REPORT DOCUMENTATION PAGE

AFRL-SR-BL-TR-98-

0153

1. AGENCY USE ONLY (Leave Blank)	2. REPORT DATE 10 Mar 97	3. REPORT TYPE AND DATES COVERED Final (01 Jan 94 – 31 Dec 96)

4. TITLE AND SUBTITLE Introspective Reasoning Models for Multistrategy Case-based and Explanation…	5. FUNDING NUMBERS F49620-94-1-0092
6. AUTHORS Professor Ashwin Ram	
7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES) Georgia Institute of Technology Atlanta, GA 30332	8. PERFORMING ORGANIZATION REPORT NUMBER
9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES) AFOSR/NM 110 Duncan Avenue, Room B-115 Bolling Air Force Base, DC 20332-8080	10. SPONSORING/MONITORING AGENCY REPORT NUMBER
11. SUPPLEMENTARY NOTES	
12a. DISTRIBUTION AVAILABILITY STATEMENT Approved for Public Release	12b. DISTRIBUTION CODE

13. ABSTRACT *(Maximum 200 words)*

On the technical front, we have been working towards using the StatLady system as the starting point for implementing our own ideas. By using StatLady, we can make use of the system's large body of domain information, its capacity for evaluating a student's comprehension of a large number of definitions, procedures, and skills related to statistics, and its tutoring algorithms. We are exploring several options for an interface, including Visual Basic, the language of the original StatLady system. On the theoretical side, we have focussed on designing our "planning to tutor" module, based on the PLUTO "planning to learn" system, for augmenting StatLady along the lines of our proposal. Our extension builds on the strengths of StatLady by expanding the way Statlady represents the student model, the way it creates a lesson plan, and the flexibility it has to adapt a lesson plan as student performance deviates from the student model's predictions.

19980205 060

14. SUBJECT TERMS Statlady, PLUTO	15. NUMBER OF PAGES
	16. PRICE CODE

17. SECURITY CLASSIFICATION OF REPORT Unclassified	18. SECURITY CLASSIFICATION OF THIS PAGE Unclassified	19. SECURITY CLASSIFICATION OF ABSTRACT Unclassified	20. LIMITATION OF ABSTRACT UL

C-36-X26
#4

MODELING MULTISTRATEGY LEARNING AS A DELIBERATIVE PROCESS OF PLANNING IN KNOWLEDGE SPACE

PRINCIPAL INVESTIGATOR:

Dr. Ashwin Ram
College of Computing
Georgia Institute of Technology
Atlanta, Georgia 30332-0280
Phone: (404) 894-4995
Fax: (404) 894-5041
E-mail: ashwin@cc.gatech.edu
Web: http://www.cc.gatech.edu/faculty/ashwin/

1. Overview

The focus of this research is a novel approach to machine learning which models learning as a goal-driven, deliberative process that uses introspection about reasoning failures to guide the planful selection and construction of learning strategies. The specific goal of this research is to encompass traditional case-based, explanation-based, and multistrategy learning approaches within a unified framework. Using this framework, agents operating in an environment can analyze their informational needs and strategically pursue those needs by constructing knowledge plans from a library of available learning algorithms and executing those plans. "Planning to learn" is closely related to the process by which agents analyze their physical needs and plan and act in the external environment. Our research has yielded a framework for explicitly representing reasoning to facilitate the detection of reasoning errors, a taxonomy of error types, a learning strategy selection algorithm to remedy various kinds of reasoning errors, a planning system which prevents learning actions from interfering with each other, and a case-base opportunistic controller that interleaves planning and plan execution in a dynamic, real-time resource-constrained environment.

2. Highlights

Our research has had an extensive impact on the machine learning community. We have published several papers in books, journals, and conferences based on the research described herein. Our *Goal-Driven Learning* book (Ram & Leake, 1995), was a major work that arose in part out of the ideas developed in this project. In addition, our approach has also been the basis for past and current dissertation projects by students outside the research group, such as Quilici's model of Unix users (in press), Redmond's model of learning via apprenticeship (1990, 1992).

and Fox's model of learning memory search strategies (1995). The pervasive influence of our approach reflects the active and wide-ranging nature of our efforts. Here are the major highlights of the research we have conducted.

Theoretical Issues

Specific theoretical contributions have been described in our annual reports; here, we summarize the main points.

- While our previous research focused on combining available learning actions, little attention was paid to the choice of learning actions themselves. We have learned that in order to develop a general theory of learning, it is important to understand the "primitives" of learning, that is, to develop a principled theory of the learning actions involved in acquiring and transforming knowledge.
- Previous research showed the need for non-linear planning in multistrategy learning systems, but we discovered several important differences between traditional planning and knowledge planning. For example, knowledge planning requires:
    - The ability to plan with incomplete information. E.g., the postconditions of a learning operator might not be known. Indeed, all postconditions may not be representable in the traditional "add-list" and "delete-list" formalism.
    - The ability to react opportunistically to unexpected situations. These opportunities may be external (e.g., a magazine article may provide information relevant to a prior learning goal) or internal (e.g., a situation may trigger an unexpected reminding).
    - The ability to deal with knowledge-level interactions that are somewhat different from the interactions found in physical plans. E.g., the familiar "brother-goal-clobbers-brother-goal" interaction in the physical world manifests itself very differently in the mental world. Understanding the task of knowledge planning, and developing a general method and supporting representations for this task, is a key objective of this project.
- While our previous research focused primarily on the learning process itself, learning is tightly integrated with reasoning and memory. We have proposed a theory of learning situated in a larger theory of agency which specifies not only the learning system but also the reasoning and memory systems and their interactions.

Technical Highlights

- We developed a multistrategy learning system called Meta-AQUA (Cox, 1996; Ram & Cox, 1994) that learns by reading stories. Meta-AQUA determines its own learning goals by analyzing its successes and failures at its task, and creates learning plans to achieve its knowledge goals; this allowed it to combine multiple learning strategies in a dynamic and flexible manner. Meta-AQUA is the first computer system that we know of that is able to create its own learning plans dynamically.
- We produced another multistrategy learning system called Meta-TS (Ram, Narayanan, & Cox, 1995) that models the performance of human troubleshooters on the assembly line of an

actual electronics manufacturing plant near Atlanta, using actual data from field studies. We showed that Meta-TS can model how human troubleshooters gradually improve their performance through experience.

- We developed a multimedia training and aiding systems for human troubleshooters, based on the theoretical results of the Meta-AQUA and Meta-TS projects. The cognitive theory of learning that we are developing was used to guide the design of the training and aiding system. The principle behind this project is that once we understand how people learn, we can use that understanding to design systems that can facilitate that learning process. The computer system, called WALTS (Minsk, Balakrishnan, & Ram, 1995) , is a multimedia workspace for aiding and training troubleshooters in an engineering domain. It will be fielded in the next few months for evaluation purposes.
- We constructed the NICOLE system (Ram & Francis, 1996): a case-based adaptive planner using asynchronous and opportunistic memory retrievals. NICOLE demonstrates how an planning agent situated in a real-time environment may need to quickly retrieve a less-than-perfect plan under time pressure, but may retrieve and merge additional plans as execution progresses. In addition, NICOLE is capable of exploiting unforeseen opportunities that appear during plan execution.
- The PLUTO system is our latest attempt in the effort to apply the power of planning to the task of learning. The current implementation of PLUTO is a simple proof-of-concept system, similar the original blocks word planners. However more ambitious versions of PLUTO are already under development.

3. The Problem: History and Issues

We have addressed two problems that are crucial in the design of intelligent agents. First, because the value of learning depends on how well the learning contributes to achieving the learner's goals, the learning process should be guided by reasoning about the information that is needed to serve those goals. Thus, computer programs need to be able to make good decisions about when and what to learn, about which algorithms to use to achieve the desired learning, and how to guide the application of the chosen strategies. Secondly, because the domains we are dealing with are dynamic and not completely understood, it is unlikely that a single prior experience or case in the agent's memory will be able to provide the necessary guidance to deal with a novel situation. Thus, computer programs need to be able to retrieve multiple cases, extract relevant pieces of these cases depending on the problem at hand, and dynamically combine these cases to create a solution to the new problem.

The research led us to consider three additional issues. First, while an introspective multistrategy learning system can incorporate arbitrary learning mechanisms, the size and computational power of these learning mechanisms is an open question. In contrast to a traditional multistrategy learning approach (in which a system has access to large, monolithic learning mechanisms like CBR and EBG, which themselves are fixed) we hypothesized that high-level learning behaviors emerge from the composition of appropriately chosen low-level operations (called by Michalski

(1993) *knowledge transmutations*). We call this the *granularity of learning* problem. Second is the problem of *deciding how to learn*: an introspective multistrategy learning system chooses which learning algorithms to apply through a process similar to planning; we hypothesized that this process could be extended to encompass the more complex planning processes used during the course of traditional planning and acting "in the world". Finally, decomposing high-level learning algorithms requires not only a representation language for knowledge transmutations but also an overall agent architecture (specifying representation language, processing assumptions, and control structures) which support a wide range of learning operations operating in concert with reasoning; we call this the *architecture for learning* problem. A solution to these three problems would constitute a novel general learning method, which would be of great benefit to learning and artificial intelligence research.

4. Our Solution: Strategy and Results

The funds provided by AFOSR were used to tackle and provide novel solutions to each of these problems. Our solutions came in the form of three separate but related bodies of work: introspective multistrategy learning, experience-based agency, and transmutation-based inference. The first key contribution of this research is the development of a new theory of learning, known as *introspective multistrategy learning*, in which the reasoning system models its own reasoning processes explicitly, and analyzes this model after a reasoning experience in order to identify what it needs to learn and to select the appropriate learning strategy from a set of available strategies. Self-modeling is accomplished using knowledge structures known as *meta-explanation patterns*. The system is able to identify its information needs—called *knowledge goals*—and pursue the necessary learning using multiple learning methods—called *introspective multistrategy learning*.

The second key contribution is the development of a new method for case-based reasoning, called *experience-based agency*, which specifies how a resource-bounded agent can deal with problems in the real world. The real world presents several challenges: it is dynamic, unpredictable, and independent, poses poorly structured problems, and places bounds on the resources available to agents. Our theory specifies how an agent with the ability to richly represent and store its experiences could remember those experiences with a context-sensitive, asynchronous memory, incorporate those experiences into its reasoning on demand with integration mechanisms, and usefully direct memory and reasoning through the use of a utility-based metacontroller. The principles of this architecture extend beyond case-based reasoning to provide support for other learning mechanisms as well.

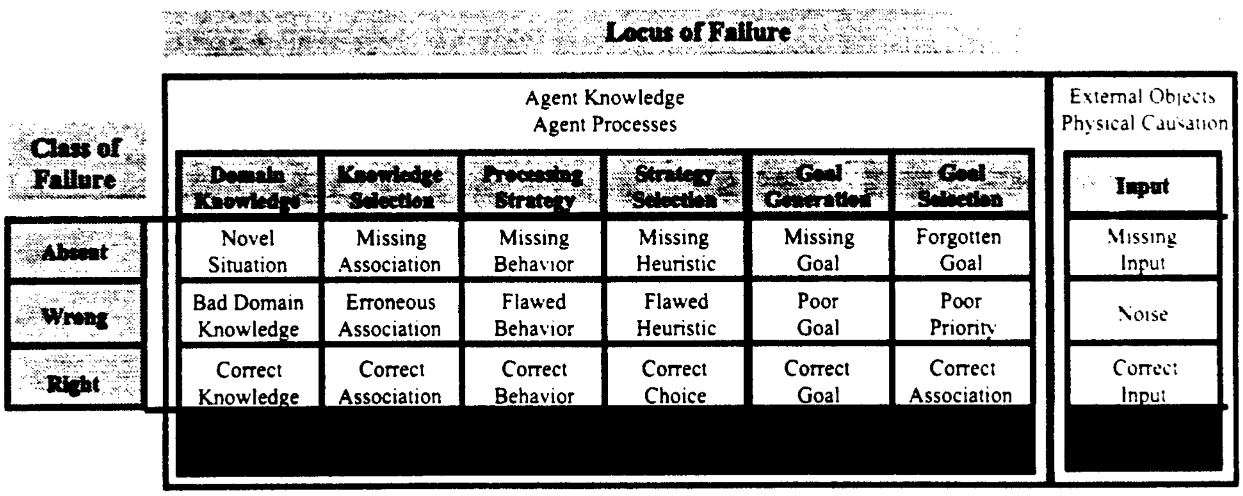
Our final key development is a framework for flexible learning called *transmutation-based inference* (or planning to learn) which specifies how an agent can satisfy its knowledge goals by composing low-level knowledge transmutations into high-level learning plans. Thus we provide a mixed approach to the granularity of learning: we developed a representation language for fine-grained knowledge transmutations in terms of production rules which operate over a knowledge base; yet these knowledge transmutations share the structure of traditional planning operators, so a system can decide how to learn by constructing plans using a wide range of planning processes.

These large-grained knowledge plans can be viewed as full-fledged learning algorithms, or as "macro" transmutations to be further composed into larger plans.

This work thus not only builds on our prior work in multistrategy learning, but also serves to unify our work in story understanding, knowledge representation, planning, learning and case-based reasoning into a cognitive architecture capable of reasoning, acting — and learning — in a wide range of domains. The enclosed technical reports provide more details on our methods and results; here, we summarize our research by presenting the underlying computational justifications of our theory and highlighting the important innovations that resulted in our work.

5. Technical Details: Theory

A central thesis of this research is that a reasoner must explicitly represent and remember the steps it takes on the way to a solution to a problem. This trace becomes essential when a failure occurs and the reasoner must diagnose and repair the failure. In Meta-AQUA, the Meta-XP holds these data. In order to adequately analyze the trace, a taxonomy of failure types provides the vocabulary needed to move to the next step: planning to learn.



	Locus of Failure						
	Agent Knowledge Agent Processes						External Objects Physical Causation
Class of Failure	Domain Knowledge	Knowledge Selection	Processing Strategy	Strategy Selection	Goal Generation	Goal Selection	Input
Absent	Novel Situation	Missing Association	Missing Behavior	Missing Heuristic	Missing Goal	Forgotten Goal	Missing Input
Wrong	Bad Domain Knowledge	Erroneous Association	Flawed Behavior	Flawed Heuristic	Poor Goal	Poor Priority	Noise
Right	Correct Knowledge	Correct Association	Correct Behavior	Correct Choice	Correct Goal	Correct Association	Correct Input

Figure 1: A taxonomy of learning failures.

If reasoning is modeled as goal-directed processing of an input using some knowledge, there is only a limited number of classes of factors that may be responsible for the success or failure of the reasoning process. A failure could stem from the goal, the process, the input, or the domain knowledge. Furthermore, if both knowledge and sets of reasoning strategies are organized in order to facilitate access to them, so that appropriate knowledge and strategies can be retrieved and brought to bear on a given situation, the organization of knowledge and reasoning strategies may be responsible for a failure as well. Finally, failures may arise from the generation, selection, and opportunistic triggering of goals as well.

Even if no failure has yet occurred, anticipation of a reasoning failure may trigger learning. For example, a reasoner may realize that it cannot perform a task and decide to perform the necessary learning before even attempting the task. In our framework, all these motivations for learning—reasoning failures, difficulties, impasses, suboptimalities, surprises, and other types of processing problems or anticipated processing problems—are collectively and simply referred to as *failures*. Reasoning processes and reasoning failures are represented using the Meta-XP representations that we have developed.

Once the reasoning faults have been identified, learning strategies that remedy the faults can be deployed. But certain learning strategies have ordering constraints. If these constraints are violated, the benefits of one or more of the learning strategies may be lost. It is necessary to represent the dependencies between learning strategies, and produce a partial ordering of the strategies that satisfies the dependencies. The act of producing a partial ordering of steps is the act of planning.

Planning to act in the world and planning to learn from some internal mental representation share deep and significant similarities. Thus, Meta-Aqua employed a conventional planning system, NONLIN, to determine a partial ordering of the learning strategies to be applied such that these learning strategies did not interfere with each other, as they might do if their order of execution was undefined.

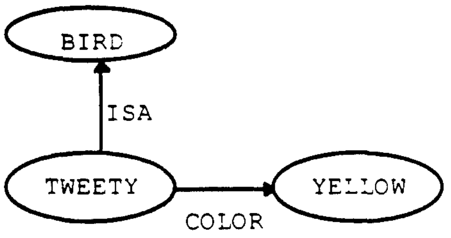
Recently, we have become interested in the power of planning to learn at lower levels of abstraction. Meta-Aqua planned at the level of learning strategies, which themselves are highly regimented plans that achieve ambitious goals in limited domains. However, it is possible to represent the underlying primitive operations that comprise learning strategies. We use Michalski's (1993) term "transmutations" to describe these generic operators. A more advanced planning to learn system could assemble customized learning strategies from transmutations on the fly, potentially creating a far larger domain of solutions. In the final stage of this project, we have built a prototype of such a system, PLUTO, and are continuing to develop it further.

5.1. Knowledge Representation for Learning Systems

Traditional monolithic learning algorithms require special-purpose knowledge representations in order to function; CBR requires a case library, while EBG requires structured representations and a domain theory. Systems that design their own learning strategies do not have this luxury; since planning to learn is a *general* method operating over a knowledge base which can approximate CBR or EBG as needed, it is important to design a general representation language which is useful enough to support a wide range of tasks, yet is rich enough to serve as a base for the planning to learn process.

Appropriate knowledge representation has been a constant thread in our research, from our initial work with the AQUA system through our current work in the NICOLE and PLUTO systems. A general knowledge representation should have enough expressive power to represent concepts in a wide range of task domains, should have a well-defined semantics to permit complex reasoning operations, and should support a powerful set of operations to manipulate the knowledge to promote the construction of complex algorithms. After extensive experimentation in the NICOLE and PLUTO systems, we hypothesize that a *multi-view, grounded relation-based* *knowledge base* meets these criteria. Since this knowledge base contains experiential knowledge as well as the more traditional semantic knowledge, it is called an "experience store".

The basic unit of this knowledge base is the concept, which are connected by links mediated by *relations*. Each instance of a relation is a tuple of the form (<relation> <domain> <codomain>): for example, an instance of the ISA relation might be (ISA TWEETY BIRD). In addition, certain privileged concepts are *grounded* in this representation by connection to nonconceptual processes: for example, the cost of a stereo might be represented by (COST SONY-5 150.0-CONCEPT), where 150.0-CONCEPT cashes out into a machine representation which can be manipulated by subcognitive arithmetic operations. Finally, a relation-based knowledge base can be viewed in several equivalent ways, each supporting different types of processing: as a set of predicates (tuples), as a semantic network (where each relation is a directed, named link connecting two nodes) and as a traditional frame representation (where all knowledge related to a specific concept is extracted from the knowledge base); Figure 1 shows these three views.



```
(ISA TWEETY BIRD)
(COLOR TWEETY YELLOW)

(FRAME TWEETY
    ISA BIRD
    COLOR YELLOW)
```

Figure 1. Three equivalent views of a relation-based knowledge representation: semantic net (left), predicate (tuple) notation (upper right), and frame notation (bottom right). From the PLUTO 2.0 specification.

We extensively tested this combination of low-level representation and high-level structure for suitability as a general knowledge representation system in the NICOLE agent architecture, which was used to construct the knowledge representation for the ISAAC (Moorman & Ram, 1994) and NICOLE-MPA (Ram & Francis, 1995) systems. However, we have extended this representation to support planning to learn in the PLUTO system in two specific ways: first, elements of tuples can be variablized; second, we have developed a standard language for creating and matching variablized templates. Together, this representation and these extensions allow us to specify not only complex knowledge relating to a wide variety of tasks, but also specify in a principled way gaps in that knowledge. This is the fundamental construct we use to represent both knowledge goals and knowledge transmutations.

5.2. Selecting and Integrating Learning Strategies

5.2.1. Knowledge Goals and Learning Strategies

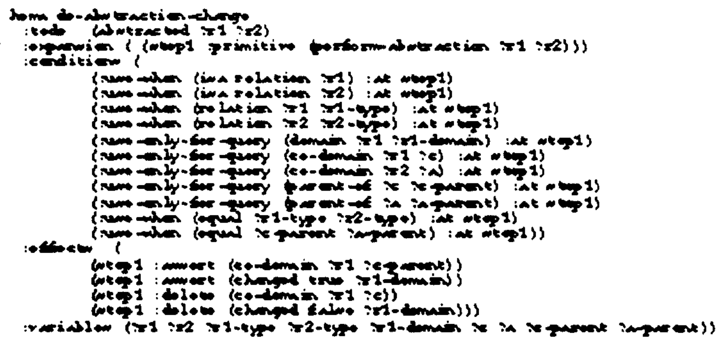
Knowledge goals (KGs) are explicit representations of gaps in a reasoner's knowledge base. These can be represented as sets of tuples where one or more elements are variables (by convention, written with a initial question mark: e.g., ?X). So for example, the canonical

"Marvin Minsky's phone number" knowledge goal might be defined by the tuple (PHONE-NUMBER MARVIN-MINSKY ?X) where ?X represents a placeholder of the information to be learned. In addition to this *knowledge specification*, the representation of a knowledge goal must also represent the *task specification*, a declaration of the suspended task that is awaiting the information that is being sought.

A knowledge goal, however it is produced, represents a challenge; once a gap has been found in the reasoner's knowledge base, to satisfy the goal the reasoner must have some mechanism for filling the gap. The basic unit of reasoning for a planning to learn system is a learning operator: a package of knowledge and/or processing elements that take various kinds of knowledge as input and produce new or rearranged knowledge as output, represented as a planning operator.

5.2.2. A Sample Learning Operator: Abstraction

In order to be able to reason about, select, and combine learning strategies in this manner, whether these are full-fledged machine learning algorithms or primitive knowledge transmutations, it is necessary to represent their applicability conditions and expected results in a declarative and explicit manner. This has a nice methodological benefit as well: it requires us to write down explicitly the conditions under which the learning algorithms we invent ought to be applied. In our previous research, we developed a formalism similar to the "preconditions add-list/delete-list" notation used in conventional non-linear planners for physical operators. For example, the figure below shows a representation of "abstraction", in which a concept in a frame system is generalized up the relation hierarchy.



```
(hems do-abstraction-change
  :todo    (abstracted ?r1 ?r2)
  :expansion ( (step1 primitive (perform-abstraction ?r1 ?r2)))
  :conditions (
        (use-when (isa relation ?r1) :at step1)
        (use-when (isa relation ?r2) :at step1)
        (use-when (relation ?r1 ?r1-type) :at step1)
        (use-when (relation ?r2 ?r2-type) :at step1)
        (use-only-for-query (domain ?r1 ?r1-domain) :at step1)
        (use-only-for-query (co-domain ?r1 ?c) :at step1)
        (use-only-for-query (co-domain ?r2 ?a) :at step1)
        (use-only-for-query (parent-of ?c ?c-parent) :at step1)
        (use-only-for-query (parent-of ?a ?a-parent) :at step1)
        (use-when (equal ?r1-type ?r2-type) :at step1)
        (use-when (equal ?c-parent ?a-parent) :at step1))
  :effects  (
        (step1 :assert (co-domain ?r1 ?c-parent))
        (step1 :assert (changed true ?r1-domain))
        (step1 :delete (co-domain ?r1 ?c))
        (step1 :delete (changed false ?r1-domain)))
  :variables (?r1 ?r2 ?r1-type ?r2-type ?r1-domain ?c ?a ?c-parent ?a-parent))
```

Figure 2. The Abstraction Operator

Note that this formalism specifies the effects of a learning operator using "assert" and "delete" clauses. This will probably be inadequate for more sophisticated learning operators; for example, "analogical mapping" results not only in the addition and deletion of facts but in the mapping of the entire structure of a model from the source to the target concept. Some operators may not have well-defined postconditions at all; for example, "read a magazine article" results in the acquisition of knowledge which is unspecified. Developing a suitable taxonomy of learning actions, and a suitable representational formalism for these actions, has been and will continue to be a key thrust of this research.

5.3. Integrating Learning Strategies within a Complete Reasoning System

A planning to learn system is only as good as the source of its inputs (the source for its knowledge goals) and the receiver of its outputs (the executive of its knowledge plans). Our goal for this research is not just to employ appropriate learning strategies, but to integrate them with the type of failure-driven trace analysis of the performance system. In Meta-AQUA and Meta-TS, we have shown that such an introspective analysis is not only feasible but also beneficial. In NICOLE, we have shown that reasoning, acting, introspecting, and learning can be integrated into a closed cycle, serving as a complete cognitive architecture based on planning to learn principles.

5.3.1. Why Planning, Knowledge Planning and Execution Must Be Integrated

There is a tight coupling between physical planning, knowledge planning, and execution. As John Pollock has pointed out (1995), achieving a knowledge goal may require the execution of arbitrarily large numbers of actions (for example, verifying the existence of the neutrino required a large multi-year research effort), whereas achieving a physical goal may require arbitrarily large amounts of knowledge and reasoning (for example, getting to the moon required a lot of brainwork on the part of NASA, even though the basic physics they tackled had been largely worked out). Furthermore, both types of goals may require interleaving planning and execution (for example, before plans for a novel space probe can be completed, it may be necessary to experiment with different materials to see which best fit the design goals, and the properties of the materials will affect the design). Thus, to achieve any goal, we may have to plan for knowledge, we may have to plan for action, and we may have to execute an action — all concurrent, interleaved, and inextricably intertwined.

5.3.2. How Planning, Knowledge Planning and Execution Can Be Integrated

It is a difficult problem to control a system that combines several learning strategies, much less the many non-learning task duties required of a performance system (such as user interaction, memory, planning, or real-world execution), especially when the strategies (and actions) must be composed into a unified plan for action. However, one of the major goals of this research is to design a control architecture capable of doing precisely that. As part of this research project, we have developed a control architecture based on the *experience-based agency* theory, implemented in the NICOLE system. An experience-based agent has five layers: a task controller, a unified knowledge representation language, a unified long term memory called an *experience store* (based on the same principles of general-purpose, multi-view relation-based knowledge used in our planning to learn work), a working memory shared by all tasks in the system, and an asynchronous memory retrieval system (a privileged task that interlocks with the other four components of the system).

From the perspective of integrating reasoning and action, the key element of an experience-based agent is a task controller capable of concurrently orchestrating the activity of several subsystems — for instance, an asynchronous memory module and a knowledge planner. The task controller combines results from task-decomposition systems, reactive systems, blackboard systems, and operating systems theory. Like Jim Firby's RAPs (1989) and Roy Turner's SBR (1989), tasks are decomposed into two parts: low-level executable actions that can be directly executed by the system, and high-level declarative specifications which can be explicitly parsed by an interactive interpreter. This breakdown imposes a hierarchical task decomposition upon the basic actions, yet allows the specifics of execution to be determined at run-time. Furthermore, it allows the control of both low-level reactive components and high-level reasoning components within a uniform framework.

An experience-based agent goes beyond RAPs and SBR in two specific ways: task communication and utility-based control. The primary organizing structure for execution knowledge is the *supertask* (which we developed in our lab under a separate project), a declarative specification of the knowledge and subtasks necessary to perform some cognitive task. A supertask defines both the working memory structures that a cognitive task uses for communication (implemented in the NICOLE system as a blackboard pane) and the set of top-level subtasks which must be active for the task to be performed. Given a set of supertasks, an experience-based agent constructs a global working memory structure from the supertask specifications and then concurrently executes the subtasks that the supertasks define.

But we need more than concurrent execution of tasks — sometimes it is necessary to choose between more than one learning method, or between attempting to perform a learning action and responding to the user or external events in a timely fashion. An experience-based agent allows utility-based competition between tasks, where two tasks (or hierarchical sets of tasks) compete for the right to execute, based on utility values computed by functions attached to the task specifications.

5.3.3. An Integrated Agent Architecture for Learning

We are now in a position to explain how our theories of learning fit into the context of a complete agent architecture. Given some agent functioning in a environment or performing some performance task, an introspective task (such as that implemented in our Meta-AQUA system) can trace reasoning episodes, analyze failures, and detect opportunities to learn in the form of knowledge goals. A planning to learn system (such as PLUTO) can analyze these knowledge goals and construct knowledge plans (which themselves may be composed from past knowledge plans retrieved from memory). Execution of these knowledge plans can be orchestrated by an overall agent architecture (such as Nicole) which interrupts the performance task with knowledge plans as necessary. Figure 4 illustrates the operation of this architecture.

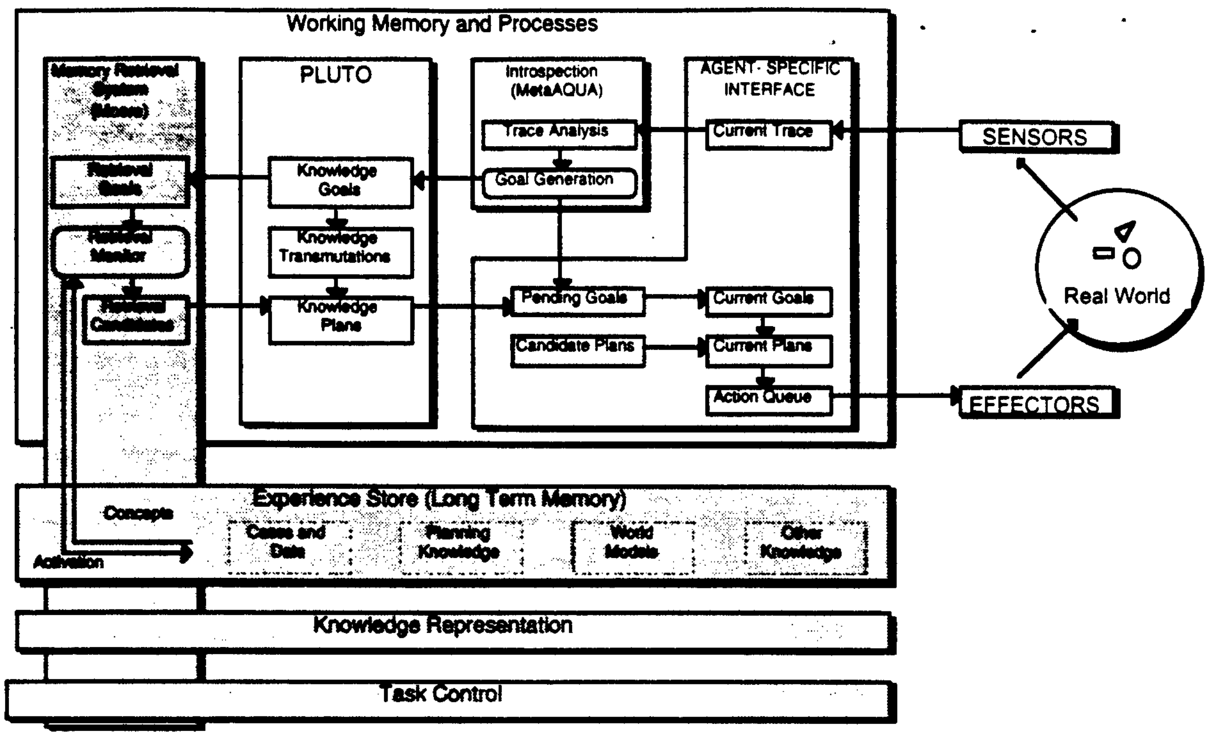


Figure 5. Planning to Learn as part of an overall Cognitive Architecture

We have separately implemented a version of the PLUTO system, the Meta-AQUA system, and the overall Nicole agent architecture (shown in gray). Integration of these components into an overall system that can function intelligently in a complex, dynamic world and learn from its experiences using multiple learning strategies will be completed as part of a future research project.

6. New Domains for Learning

To show that our theories have generality beyond any one domain, or for only a few hand-coded situations, we have deployed our theories into diverse, complex domains. Meta-AQUA reads simple natural language stories and learns from them. Meta-TS models human troubleshooters on an electronics assembly line. NICOLE solve planning problems. Details are provided in the attached publications. We list below two additional domains which we have begun to explore.

6.1. Consumer Decision-Making

Consumer decision-making is a learning and information gathering process involving multiple learning strategies. Buyers are faced with external constraints, such as how much money they have to spend or how much time and trouble they are willing to invest to make sure that they make a good decision. Buyers also have internal constraints: they have imperfect knowledge of the range and availability of products, the attributes of the products such as reliability or quality, the relevance of various attributes to their specific purpose in buying the product, and so on.

Finally, buyers can make a many different inferences, rangingfrom explanation-based analysis of product attributes and performance to inductive generalization over different products.

In order to incorporate a larger body of real-world data, we obtained ratings for a few dozen models of consumer products from Consumer Reports. Using vacuum cleaner ratings, we observed what people do when faced with learning goals identical to those faced by the PLUTO system. Four voluntary participants viewed the Consumer Reports data in its original, tabular form. We posed several questions to our subjects. Using a spoken aloud protocol, we traced their reasoning processes in order to learn what sort of strategies and transformations subjects applied to the problems. Our current prototype of the PLUTO system is aimed at modeling these reasoning and learning behaviors.

6.2. Intelligent Teaching Systems (StatLady)

A very different application of planning to learn is in the domain of instruction. Intelligent teaching systems (ITSs) have been a long-time area of interest for artificial intelligence researchers. The concepts behind PLUTO are particularly applicable because to be an effective teacher, an instructor must also be an effective learner. In order to deliver proper instruction, an instructor must assess the skills and knowledge of his student. The instructor should know what the student does and does not understand, and the relationship between this partial understanding and the full level of expertise that the instructor intends to teach. Only given an accurate model of the student's understanding can an instructor create an optimal lesson plan. The task of gauging learner proficiency is complicated by the fact that students can learn or forget material of their own accord. Also, excessive testing detracts from the learners' time and attention, slowing the overall learning process. The utility of testing must be taken into account, and each evaluation should yield the maximum amount of information relative to its cost.

This sort of intelligent student evaluation can be thought of as a special form of planning to learn: planning to diagnose. Planning to diagnose can occur at several places. First, a new student may require significant evaluation to construct an initial model of their competence. Second, this model must be maintained and updated as a student learns. Third, if a student unexpectedly fails at an advanced skill, prerequisite skills must be evaluated to find the cause of the failure and suggest a remedy.

Last, but not least, some students use significantly different learning strategies than other students. Discovering which strategy an individual student favors may suggest which sorts of instruction will prove most fruitful.

6.3. Agent Simulation (PHOENIX)

It is extremely difficult to evaluate an overall cognitive architecture because systems such as NICOLE are capable of not only *combining* planning, action, and learning but also *exploiting* the combination for improved performance. For example, such an architecture should be able to use the data gathered during plan execution to guide memory retrieval and to streamline explicit knowledge planning. Unfortunately, traditional planning tasks do not allow us to exploit this channel of communication for the simple reason that classical planning explicitly separates planning and action, making "concurrent execution" a poorly defined concept.

From the machine learning side, it is easy to run individual machine learning algorithms on traditional datasets, but this does not tell us much about how to evaluate sophisticated multistrategy learning algorithms which allow systems to learn not only *from* experience but also *during* experience. In fact, few machine learning algorithms have any experiences at all, unless they are embedded in larger systems which do something — planning, design, diagnosis, and so on.

We believe strongly in developing machine learning algorithms in the context of intelligent systems — systems which perform intelligent tasks in realistic environments. For this reason, we have turned to interactive simulators, such as PHOENIX (Cohen et al. 1988), which require both sophisticated planning and execution in realistic environments. By "realistic" we mean that the environment possesses several key features: the world is not completely under the control of the intelligent system; events happen in (simulated) real-time; the system is performing some complex task; and knowledge goals arise in support of these tasks, making learning necessary in a principled manner.

As part of this project, we have modified the PHOENIX simulator to interface with the NICOLE agent architecture. In addition to providing a more rigorous testbed for the integration of planning and acting in the agent architecture itself, this realistic environment provides greater opportunities and challenges for the planning to learn component. By comparing PLUTO's capability to devise learning mechanisms on the fly with special-purpose learning algorithms designed to cope with the PHOENIX environment, we will be able to evaluate whether the planning to learn paradigm will scale up to realistic tasks.

7. Conclusion

In conclusion, we have developed a computational model of learning with the following properties: (i) learners model their own reasoning processes for the purpose of detecting reasoning failures; (ii) a taxonomy of possible reasoning failures guides the reasoner in selecting and combining one or more strategies from a library of learning strategies; (iii) nonlinear planning helps the reasoner avoid the pitfalls of learning strategy; (iv) learning is a deliberative, planful process: knowledge plans are composed through processes paralleling traditional planning; (v) learning is integrated with reasoning: the knowledge representations and processing structures used for learning are identical to those used in an agent's overall reasoning and acting architectures. Our theory provides the foundation for a novel cognitive architecture for a multistrategy learning system. We are grateful to AFOSR for providing the funding for this research effort.

8. Publications

Our research has resulted in one completed PhD thesis (Michael Cox), one nearly-completed PhD thesis (Anthony Francis), an edited volume (Ashwin Ram & David Leake), and several publications. Here is a complete list of publications produced under this grant. Copies of last year's publications are included with this report; copies of earlier publications have been provided with previous annual reports.

- M.T. Cox (1996). Introspective Multistrategy Learning: Constructing a learning strategy under reasoning failure. PhD dissertation, Technical Report GIT-CC-96/06, College of Computing, Georgia Institute of Technology, Atlanta, GA.
- M.T. Cox (1996). An Empirical Study of Computational Introspection: Evaluating introspective multistrategy learning in the Meta-AQUA system, *Third International Workshop on Multistrategy Learning*, 135–146, AAAI Press, Menlo Park, CA.
- D.B. Leake & A. Ram (1996). Learning, Goals, and Learning Goals: A Perspective on Goal-Driven Learning, *Artificial Intelligence Review*.
- A. Ram & A. Francis (1996), Multi-Plan Retrieval and Adaptation in an Experience-Based Agent, in *Case-Based Reasoning: Experiences, Lessons, and Future Directions*, D.B. Leake, editor, AAAI Press.
- M.T. Cox & A. Ram (1995). Interacting Learning-Goals: Treating Learning as a Planning Task. In J.-P. Haton, M. Keane, & M. Manago (eds.), *Topics in Case-Based Reasoning* (Lecture Notes in Artificial Intelligence), 60-74, Springer-Verlag, 1995
- M.T. Cox & A. Ram (1995). Managing Learning Goals in Strategy Selection Problems. In *Topics in Case-Based Reasoning: Second European Workshop*, M. Keane, J.-P. Haton, & M. Manago, editors, 60–74, Springer-Verlag.
- D. Aha & A. Ram, editors (1995). *Proceedings of the AAAI Fall Symposium on Adaptation of Knowledge for Reuse*, AAAI Press, Menlo Park, CA.
- M.T. Cox & M. Freed, editors (1995). *Proceedings of the 1995 AAAI Spring Symposium on Representing Mental States and Mechanisms*, Technical Report SS-95-08, AAAI Press, Menlo Park, CA.
- M.T. Cox (1995). Representing mental events (or the lack thereof). In *AAAI Spring Symposium on Representing Mental States and Mechanisms*, Stanford, CA.
- A.G. Francis & A. Ram (1995). A Comparative Utility Analysis of Case-Based Reasoning and Control-Rule Learning Systems, *Eighth European Conference on Machine Learning*, Crete, Greece.
- A.G. Francis & A. Ram (1995). An Algorithm for Multi-Plan Adaptation and Merging for Least-Commitment Planners, *AAAI Fall Symposium on Adaptation of Knowledge for Reuse*, Boston, MA.
- B. Minsk, H. Balakrishnan, & A. Ram (1995). Structuring On-The-Job Troubleshooting Performance to Aid Learning. In *World Conference on Engineering Education*, Minneapolis, MN.
- A. Ram, S. Narayanan, & M.T. Cox (1995). Learning to Troubleshoot: Multistrategy Learning of Diagnostic Knowledge for a Real-World Problem Solving Task. *Cognitive Science* journal, 19(3):289-340.
- A. Ram & D. Leake, editors (1995). *Goal-Driven Learning*, MIT Press/Bradford Books. Includes the following chapters by our group:
  - A. Ram & D. Leake. Learning, Goals, and Learning-Goals.
  - A. Ram, M.T. Cox, & S. Narayanan. Goal-Driven Learning in Multistrategy Reasoning and Learning Systems.
  - R.S. Michalski & A. Ram. Machine Learning as Goal-Driven Inference.
  - A. Ram & L.E. Hunter. The Use of Explicit Goals for Knowledge to Guide Inference and Learning
  - D.B. Leake & A. Ram. Goal-Driven Learning: Fundamental Issues and Symposium Report.

- A. Ram, S. Narayanan, & M.T. Cox (1995). Learning to Troubleshoot: Multistrategy Learning of Diagnostic Knowledge for a Real-World Problem Solving Task. *Cognitive Science*, 19(3):289–340.
- M.T. Cox (1994). Machines that Forget: Learning from Retrieval Failure of Mis-Indexed Explanations. *Sixteenth Annual Conference of the Cognitive Science Society*, Atlanta, GA.
- M.T. Cox & A. Ram (1994). Choosing Learning Strategies to Achieve Learning Goals. *AAAI Spring Symposium on Goal-Driven Learning*, 12-21, Stanford, CA.
- M. T. Cox, A Ram. (1994). Managing Learning Goals in Strategy Selection Problems. *Second European Workshop on Case-Based Reasoning*, Chantilly, France.
- M.T. Cox & A. Ram (1994). Failure-Driven Learning as Input Bias. *Sixteenth Annual Conference of the Cognitive Science Society*, Atlanta, GA.
- A. Ram & M.T. Cox (1994). Introspective Reasoning using Meta-Explanations for Multistrategy Learning. In *Machine Learning A Multistrategy Approach*, Vol. IV, R.S. Michalski and G. Tecuci (eds.), 349-377, Morgan Kaufmann.
- A. Ram & D.B. Leake (1994). A Framework for Goal-Driven Learning. *AAAI Spring Symposium on Goal-Driven Learning*, Stanford, CA.
- D.B. Leake & A. Ram (1993). Goal-Driven Learning: Fundamental Issues and Symposium Report. *AI Magazine*, 14(4):67-72.
- M.T. Cox & A. Ram (1992). An Explicit Representation of Forgetting. *Sixth International Conference on Systems Research, Informatics and Cybernetics*. Baden-Baden, Germany.
- M.T. Cox & A. Ram. (1992). Multistrategy Learning with Introspective Meta-Explanations. *Machine Learning: Ninth International Conference*, Aberdeen, Scotland.
- S. Narayanan & A. Ram (1992). Learning to Troubleshoot in Electronics Assembly Manufacturing. Ninth International Conference, *Workshop on Integrated Learning in Real-world Domains*, Aberdeen, Scotland.
- S. Narayanan, A. Ram, S.M. Cohen, C.M. Mitchell, T. Govindraj (1992). Knowledge-Based Diagnostic Problem Solving and Learning in the Test Area of Electronics Assembly Manufacturing. *SPIE Symposium on Applications of AI X: Knowledge-Based Systems*, Orlando, FL.
- A. Ram, M.T. Cox, & S. Narayanan (1992). An Architecture for Integrated Introspective Learning Machine Learning: *Ninth International Conference, Workshop on Computational Architectures*, Aberdeen, Scotland.
- M.T. Cox & A. Ram (1991). Using Introspective Reasoning to Select Learning Strategies. *First International Workshop on Multistrategy Learning*, 217-230, Harpers Ferry, WV.

9. References and Related Readings

Aamodt, A. (1991). *A Knowledge Intensive Approach to Problem-Solving and Sustained Learning* . PhD Dissertation. University of Trondheim, Norwegian Institute of Technology, May, 1991.

Anderson, John R. (1983). *The Architecture of Cognition*. Cambridge, Massachusetts: Harvard University Press.

Barsalou, L. (1991). Deriving Categories to Achieve Goals. In G. H. Bower, editor, *The Psychology of Learning and Motivation: Advances in Research and Theory*, Volume 27, Academic Press, New York, NY.

Birnbaum, L. (1986). *Integrated Processing in Planning and Understanding*. PhD Thesis, Research Report 468, Department of Computer Science, Yale University, New Haven, CT.

Birnbaum, L., Collins, G., Freed, M., & Krulwich, B. (1990). Model-Based Diagnosis of Planning Failures. In *Proceedings of the Eighth National Conference on Artificial Intelligence*, pages 318–323, Boston, MA.

Brachman, R. J. (1985). An Overview of the KL-ONE Knowledge Representation System. Cognitive Science. 9, (1985) 171-216

Bratman, M. (1987). *Intentions, Plans and Practical Reason*. Harvard.

Carbonell, J. G. (1986). Derivational Analogy: A Theory of Reconstructive Problem Solving and Expertise Acquisition. In R. S. Michalski, J. G. Carbonell, & T. M. Mitchell, editors, *Machine Learning II: An Artificial Intelligence Approach*, pages 371–392, Morgan Kaufman Publishers, San Mateo, CA.

Chien, S. (1989). Using and Refining Simplifications: Explanation-based Learning of Plans in Intractable Domains. In *Proceedings of the Eleventh International Joint Conference on Artificial Intelligence*, pages 590-595, Detroit, MI.

Cohen, P. R., & Howe, A. E. (1988). How Evaluation Guides AI Research. *AI Magazine*, 9(4):35–43.

Cohen, P.R., Greenberg, M.L., Hart, D.M., &Howe, A.E. (1989). Trial by fire: understanding the design requirements for agents in complex environments. AI Magazine, 10(3): 32-48.

Collins, G. & Birnbaum, L. (1988). An Explanation-Based Approach to the Transfer of Planning Knowledge Across Domains. In *Proceedings of the 1988 AAAI Spring Symposium on Explanation-based Learning*, pages 107–111, Stanford, CA.

Cox, M. T. & Ram, A. (1994). Managing Learning Goals in Strategy Selection Problems. In *Proceedings of the Second European Workshop on Case-Based Reasoning*, pages 85–93, Chantilly, France.

Cox, M. T. (1996). *Introspective multistrategy learning: Constructing a learning strategy under reasoning failure*. PhD Thesis, Technical Report GIT-CC-96/06, College of Computing, Georgia Institute of Technology, Atlanta, GA.

desJardins, M. (1992). Goal-Directed Learning: A Decision-Theoretic Model for Deciding What to Learn Next. In *Proceedings of the ML-92 Workshop on Machine Discovery*, pages 147–151, Ninth International Machine Learning Conference, University of Aberdeen, Scotland.

Etzioni, O., Hanks, S., Weld, D., Draper, D., Lesh, N., & Williamson, M. (1992). An Approach to Planning with Incomplete Information. In *Proceedings of the International Conference on Knowledge Representation*, Morgan Kaufmann, San Francisco, CA.

Faries, J. & Reiser, B. (1988). Access and Use of Previous Solutions in a Problem Solving Situation. In *Proceedings of the Tenth Annual Conference of the Cognitive Science Society*, pages 433–439, Montreal.

Fikes, Hart and Nilsson. (1972). STRIPS. *Artificial Intelligence*, 1972, pp. 189-208.

Firby, J. (1989). Adaptive Execution in Complex Dynamic Worlds Ph.D. Thesis, Yale University Technical Report, YALEU/CSD/RR #672, January 1989.

Fox, S. (1995). Introspective Learning for Case-Based Reasoning. Ph.D. Thesis, Indiana University, Department of Computer Science, 1995.

Francis, A. and Ram, A. (1995). A Comparative Utility Analysis of Case-Based Reasoning and Control-Rule Learning Systems. In *Proceedings, ECML-95*, Heraklion, Crete, 1995.

Francis, A. G., & Ram, A. (1995). A Comparative Utility Analysis of Case-Based Reasoning and Control-Rule Learning Systems, in *Proceedings of the Eighth European Conference on Machine Learning*, 1995.

Francis, A.G. (1995). "Sibling Rivalry." *The Leading Edge: Magazine of Science Fiction and Fantasy*, 30, February 1995, pp. 79-102.

Freed, M. & Collins, G. (1993). A Model-Based Approach to Learning from Attention-Focusing Failures. In *Proceedings of the Fifteenth Annual Conference of the Cognitive Science Society*, pages 434–439, Boulder, CO.

Goel, A.K., Ali, K.S., Donnellan, M.W., Garza, A.G., and Callantine, T.J. (1994). Multistrategy Adaptive Path Planning. *IEEE Expert (9) 6*, December 1994, pp. 57-65.

Golden, K., Etzioni, O., & Weld, D. (1994). Omnipotence Without Omniscience: Sensor Management in Planning. In *Proceedings of AAAI-94*. AAAI Press MIT Press.

Graesser, A. C., Person, N., & Huber, J. (1992). Mechanisms that Generate Questions. In T. W. Lauer, E. Peacock, and A. C. Graesser, editors, *Questions and Information Systems*, pages 167–187, Lawrence Erlbaum Associates, Hillsdale, NJ.

Gratch, J. & DeJong, G. (1993). Assessing the Value of Information to Guide Learning Systems. In *Proceedings of the ML-93 Workshop on Knowledge Compilation and Speedup Learning*, pages 65–71, Tenth International Machine Learning Conference, University of Massachusetts, Amherst, MA.

Gratch, J., DeJong, G., & Chien, S. A. (1994). Deciding When and How to Learn. In M. desJardins & A. Ram, editors, *Proceedings of the AAAI Spring Symposium on Goal-Driven Learning*, pages 36–45, Stanford, CA.

Greeno, J. G. & Simon, H. A. (1988). Problem Solving and Reasoning. In R. C. Atkinson, H. Hernstein, G. Lindzey, & R. D. Luce, editors, *Stevens' Handbook of Experimental Psychology, Vol. 2: Learning and Cognition*, pages 589–673, Wiley, New York.

Hadzikadic, M. & Yun, D. Y. Y. (1988). Concept Formation by Goal-Driven, Context-Dependent Classification. In Z. W. Ras & L. Saitta, editors, *Methodologies for Intelligent Systems*, 3:322–332, 1988.

Hammond, K. (1989). Opportunistic Memory. In *Proceedings of the 1989 Meeting of the International Joint Committee on Artificial Intelligence*.

Hammond, K. J. (1989). *Case-Based Planning: Viewing Planning as a Memory Task*. Academic Press, Boston, MA.

Hammond. K., Converse, T., Marks, M., & Seifert, C. M. (1993). Opportunism and Learning. *Machine Learning*, 10(3):279–309.

Hanks, S., & Weld, D. (1995). A Domain-Independent Algorithm for Plan Adaptation. *Journal of Artificial Intelligence Research 2*, pp. 319-360.

Hayes-Roth, B. (1995). Agents on stage: Advancing the State of the Art of AI. *Knowledge Systems Laboratory Report No. KSL 95-50.* May 1995, Knowledge Systems Laboratory, Stanford University.

Hayes-Roth, B., & Hayes-Roth, F. (1979). A cognitive model of planning. *Cognitive Science (2),* pp. 275-310.

Hayes-Roth, F. & Lesser, V. (1976). Focus of attention in a distributed logic speech understanding system. In *Proceedings of the IEEE International Conference on Accoustics, Speech and Signal Processing*, pages 416–420, Philadelphia, PA.

Hinrichs, T. (1992). *Problem Solving in Open Worlds: A Case Study in Design.* Lawrence Erlbaum Associates, Hillsdale, NJ.

Hoffman, C., Mischel, W., & Mazze, K. (1981). The Role of Purpose in the Organization of Information about Behavior: Trait-based versus Goal-Based Categories in Human Cognition. *Journal of Personality and Social Psychology*, 39:211–255.

Hunter, L. E. (1990). Planning to Learn. In *Proceedings of the Twelfth Annual Conference of the Cognitive Science Society*, pages 261–276, Boston, MA.

Klimesch, W. J. (1994). *The structure of long-term memory: A connectivity model of semantic processing.* LEA.

Kocabas, S. (1994). Goal-Directed Discovery and Explanation in Particle Physics. In M. desJardins & A. Ram, editors, *Proceedings of the AAAI Spring Symposium on Goal-Driven Learning*, pages 54–61, Stanford, CA.

Kolodner, J.L. (1993). *Case-based Reasoning.* Morgan Kaufmann, 1993.

Krulwich, B., Birnbaum, L., & Collins, G. (1992). Learning Several Lessons from one Experience. In *Proceedings of the Fourteenth Annual Conference of the Cognitive Science Society*, pages 242–247, Bloomington, IN.

Laird, J. E. (1991), editor. Special Section on Integrated Cognitive Architectures. *SIGART Bulletin*, 2(4):12–184.

Laird, J. E., Rosenbloom, P. S., & Newell, A. (1986). Chunking in Soar: The Anatomy of a General Learning Mechanism. *Machine Learning*, 1:11–46.

Leake, D. B. (1992). *Evaluating Explanations: A Content Theory*. Lawrence Erlbaum Associates, Hillsdale, NJ.

Leake, D. B. (1993). Learning Adaptation Strategies by Introspective Reasoning about Memory Search. In D. B. Leake, editor, *Proceedings of the AAAI-93 Workshop on Case-Based Reasoning*, pages 57–63, AAAI Press, Menlo Park, CA.

Leake, D. B. (1995). Goal-Driven Integration of Explanation and Action. In A. Ram & D. B. Leake, editors, *Goal-Driven Learning*, MIT Press/Bradford Books, Cambridge, MA.

Maes, P. (1990). Situated agents can have goals. In. P. Maes, Ed., *Designing Autonomous Agents: Theory and Practice from Biology and Engineering and Back.* MIT.

Michalski, R. S. & Ram, A. (1995). Learning as Goal-Driven Inference. In A. Ram & D. B. Leake, editors, *Goal-Driven Learning*, Chapter 21, MIT Press/Bradford Books, Cambridge, MA.

Michalski, R. S. (1993). Inferential Theory of Learning as a Conceptual Basis for Multistrategy Learning. *Machine Learning*, 11(2/3):111–151.

Mitchell, T. M. & Keller, R. (1983). Goal Directed Learning. In *Proceedings of the International Machine Learning Workshop*, pages 117–118, Monticello, IL.

Mooney, R. J. & Ourston, D. (1993). A Multistrategy Approach to Theory Refinement. In R. S. Michalski & G. Tecuci, editors, *Machine Learning: A Multistrategy Approach, Volume IV*, pages 141–164, Morgan Kaufman Publishers, San Mateo, CA.

Moorman, K. & Ram, A. (1994). Integrating Creativity and Reading: A Functional Approach. In *Proceedings of the Sixteenth Annual Conference of the Cognitive Science Society*, Atlanta, GA, August 1994.

Mostow, J. & Bhatnagar, N. (1987). FAILSAFE—A Floor Planner that Uses EBG to Learn from its Failures. In *Proceedings of the Tenth International Joint Conference on Artificial Intelligence*, pages 249–255, Milan, Italy.

Newell, A. & Simon, H. (1972). *Human Problem Solving.* Prentice-Hall, Englewood Cliffs, NJ.

Newell, A. (1990). *Unified Theories of Cognition.* Harvard University Press. Cambridge, MA.

Ng, E. & Bereiter, C. (1991). Three Levels of Goal Orientation in Learning. *The Journal of the Learning Sciences*, 1(3&4):243–271.

Nilsson, N. (1995). Eye on the prize. *AI Magazine (16) 2*, Summer 1995, pp.9-17.

Orasanu, J. & Connolly, T. (1993). The Reinvention of Decision Making. In Klein, G. A., Orasanu, J., Calderwood, R., and Zsambok, C.E., eds., *Decision Making in Action: Models and Methods.* Ablex.

Owens, C. (1991). A Functional Taxonomy of Abstract Plan Failures. In *Proceedings of the Thirteenth Annual Conference of the Cognitive Science Society*, pages 167–172, Chicago, IL.

Park, Y. & Wilkins, D. (1990). Establishing the Coherence of an Explanation to Improve Refinement of an Incomplete Knowledge Base. In *Proceedings of the Eighth National Conference on Artificial Intelligence*, pages 511-516, Boston, MA.

Patalano, A., Seifert, C., & Hammond, K. (1993). Predictive Encoding: Planning for Opportunities. *Proceedings of the Fifteenth Annual Conference of the Cognitive Science Society*, pages 800–805, Boulder, CO.

Plaza, E., Aamodt, A., Ram, A., Van de Welde, W., & van Someren, M. (1993). Integrated Learning Architectures. In *Proceedings of the European Conference on Machine Learning*, Vienna, Austria.

Pollock, J.L. (1995). *Cognitive Carpentry: A blueprint for how to build a person*. MIT Press.

Provost, F. J. (1994). Goal-Directed Inductive Learning: Trading off Accuracy for Reduced Error Cost. In M. desJardins & A. Ram, editors, *Proceedings of the AAAI Spring Symposium on Goal-Driven Learning*, pages 94–101, Stanford, CA.

Pryor, L. & Collins, G. (1992). Planning to Perceive. In *Proceedings of the 1992 AAAI Spring Symposium on Selective Perception*, Stanford, CA.

Quilici, A. (in press). Toward Automatic Acquisition of an Advisory System's Knowledge Base. *Applied Intelligence*.

Ram, A. & Cox, M. (1994). Choosing Learning Strategies to Achieve Learning Goals. In *Proceedings of the AAAI Spring Symposium on Goal-Driven Learning*, Stanford, CA, 1994

Ram, A. & Cox, M. T. (1994). Introspective Reasoning using Meta-Explanations for Multistrategy Learning. In R. S. Michalski & G. Tecuci, editors, *Machine Learning: A Multistrategy Approach, Volume IV*, pages 349–377, Morgan Kaufman Publishers, San Mateo, CA.

Ram, A. & Hunter, L. (1992) The Use of Explicit Goals for Knowledge to Guide Inference and Learning. *Applied Intelligence*, 2(1):47-73.

Ram, A. (1987). AQUA: Asking Questions and Understanding Answers. In *Proceedings of the Sixth Annual National Conference on Artificial Intelligence*, pages 312–316, Seattle, WA.

Ram, A. (1989). *Question-Driven Understanding: An Integrated Theory of Story Understanding, Memory and Learning*. Ph. D. Dissertation, Research Report #710, Yale University, Department of Computer Science, New Haven, CT.

Ram, A. (1990). Knowledge Goals: A Theory of Interestingness. In *Proceedings of the Twelfth Annual Conference of the Cognitive Science Society*, pages 206–214, Cambridge, MA.

Ram, A. (1991). A Theory of Questions and Question Asking. *The Journal of the Learning Sciences*, 1(3&4):273–318.

Ram, A. (1993). Indexing, Elaboration and Refinement: Incremental Learning of Explanatory Cases. *Machine Learning*, 10:201–248.

Ram, A., & Francis, A. G. (1996). Multi-Plan Retrieval and Adaptation in an Experience-Based Agent, in D. B. Leake, editor, *Case-Based Reasoning: Experiences, Lessons, and Future Directions*, AAAI Press.

Ram, A., & Leake, D. B. (1995). Learning, Goals, and Learning Goals. In A. Ram & D. B. Leake, editors, *Goal-Driven Learning*, Chapter 1, MIT Press/Bradford Books, Cambridge, MA.

Ram, A., Cox, M. T., & Narayanan, S. (1995). Goal-Driven Learning in Multistrategy Reasoning and Learning Systems. In A. Ram & D. B. Leake, editors, *Goal-Driven Learning*, Chapter 16, MIT Press/Bradford Books, Cambridge, MA.

Ram, A., Narayanan, S., & Cox, M. T. (1995). Learning to Troubleshoot: Multistrategy Learning of Diagnostic Knowledge for a Real-World Problem Solving Task. *Cognitive Science*, 19(3):289–340.

Redmond, M. (1990). Distributed cases for case-based reasoning: Facilitating use of multiple cases. In Proceedings of AAAI-90. Cambridge, MA: AAAI Press/MIT Press.

Redmond, M. (1992). *Learning by Observing and Understanding Expert Problem Solving*. Ph. D. Dissertation. Technical report GIT-CC-92/43, College of Computing, Georgia Institute of Technology, Atlanta, GA.

Riesbeck, C. (1981). Failure-driven Reminding for Incremental Learning. In *Proceedings of the Seventh International Joint Conference on Artificial Intelligence*, pages 115–120, Vancouver, British Columbia.

Riesbeck, C. (1993) Replacing CBR: Now What? Invited talk. *AAAI-93 Workshop on Case-Based Reasoning*. Washington, DC, July.

Russel, S., & Norvig, P. (1995). *Artificial intelligence: A modern approach*. Morgan Kaufmann.

Schank, R. C. & Abelson, R. (1977). *Scripts, Plans, Goals and Understanding*. Lawrence Erlbaum Associates, Hillsdale, NJ.

Schank, R. C. & Leake, D. B. (1989). Creativity and Learning in a Case-Based Explainer. *Artificial Intelligence*, 40(1-3):353–385. Also appears in J. G. Carbonell, editor, *Machine Learning: Paradigms and Methods*, MIT Press, Cambridge, MA, 1990.

Schank, R. C. (1982). *Dynamic Memory: A Theory of Learning in Computers and People*. Cambridge University Press, Cambridge, England.

Schank, R. C. (1986). *Explanation Patterns: Understanding Mechanically and Creatively*. Lawrence Erlbaum Associates, Hillsdale, NJ.

Schlimmer, J. C. (1987). Incremental Adjustment of Representations for Learning. In *Proceedings of the Fourth International Workshop on Machine Learning*, pages 79–90, Irvine, CA.

Seifert, C. (1988). Goals in Reminding. In J. L. Kolodner, editor, *Proceedings of the Case-Based Reasoning Workshop*, pages 190–208, Morgan Kaufmann Publishers, San Mateo, CA.

Simina, M. & Kolodner, J. (1995). Opportunistic Reasoning: A Design Perspective. In *Proceedings of the 17th Annual Cognitive Science Conference*. Pittsburg, PA, July 1995.

Smyth, B. & Keane, M. (1995). Remembering to Forget: A Competence-Preserving Deletion Policy for CBR Systems. In *Proceedings of IJCAI-95*.

Sperber, D. & Wilson, D. (1986). *Relevance: Communication and Cognition*. Language and Thought Series, Harvard University Press, Cambridge, MA.

Srull, T. & Wyer, R. (1986). The Role of Chronic and Temporary Goals in Social Information Processing. In R. Sorrentino and E. Higgins, editors, *Handbook of Motivation and Cognition: Foundations of Social Behavior*, pages 503–549. Guilford Press, Guilford, CT.

Steinbart, P. J. (1992). The Role of Questioning in Learning from Computer-Based Decision Aids. In T. W. Lauer, E. Peacock, and A. C. Graesser, editors, *Questions and Information Systems*, pages 273–285, Lawrence Erlbaum Associates, Hillsdale, NJ.

Sternberg, R. J. (1985). Teaching Critical Thinking: Are We Making Critical Mistakes? *Phi Delta Kappa*, November 1985, p194-198.

Sternberg, R. J. (1986). *Intelligence Applied: Understanding and Increasing Your Intellectual Skills*. Harcourt, Brace, and Jovonovitch.

Stroulia, E. & Goel, A. K. (1994). Task Structures: A Language for Learning Goals. In M. desJardins & A. Ram, editors, *Proceedings of the AAAI Spring Symposium on Goal-Driven Learning*, pages 112–121, Stanford, CA.

Stroulia, E., Shankar, M., Goel, A. K., & Penberthy, L. (1992). A Model-Based Approach to Blame Assignment in Design. In J. S. Gero, editor, *Proceedings of the Second International Conference on AI in Design*, pages 519–537.

Sussman, G. (1975). *A Computer Model of Skill Acquisition*. Artificial Intelligence Series, Volume 1. American Elsevier, New York, NY.

Tambe, M., Newell, A., & Rosenbloom, P. S. (1990). The Problem of Expensive Chunks and its Solution by Restricting Expressiveness. *Machine Learning*, 5:299-348.

Turner, R. (1989). A schema-based model of adaptive problem solving. Ph.D. diss., *GIT-ICS-89/42*., Department of Information and Computer Science, Georgia Tech.

VanLehn, K. (1989). Problem Solving and Cognitive Skill Acquisition. In M. I. Posner, editor, *Foundations of Cognitive Science*, pages 527–579, MIT Press, Cambridge, MA.

VanLehn, K. (1991a). Rule Acquisition Events in the Discovery of Problem Solving Strategies. *Cognitive Science*, 15(1):1–47.

VanLehn, K. (1991b). *Architectures for Intelligence: The Twenty-Second Carnegie-Mellon Symposium on Cognition*. Lawrence Erlbaum Associates, Hillsdale, NJ.

Veloso, M. (1995). *Planning and Learning by Analogical Reasoning*. Springer-Verlag, 1995.

Weintraub, M. (1991). *An Explanation-Based Approach to Assigning Credit*, Ph. D. Dissertation, Computer Science Department, The Ohio State University, Columbus, OH.

Weld, D. (1994). An introduction to least-commitment planning. *AI Magazine, (15) 4*, Winter 1994, pages 27-61.

Wilensky, R (1986). Some Problems and Proposals for Knowledge Representation *Technical Report #UCB CSD 86/294* Dept. of Electrical Engineering & Computer Science, University of California - Berkeley.

Wilensky, R. (1983). *Planning and Understanding*. Addison-Wesley, Reading, MA.

Wisniewski, E. J. & Medin, D. L. (1991). Harpoons and Long Sticks: The Interaction of Theory and Similarity in Rule Induction. In D. Fisher & M. J. Pazzani, editors, *Concept Formation: Knowledge and Experience in Unsupervised Learning*, Morgan Kaufman Publishers, San Mateo, CA.

Wooldridge, M. & Jennings, N. (1995) Intelligent Agents: Theory and Practice. Submitted to *Knowledge Engineering Review* October 1994, Revised January 1995.

Xia, X. & Yeung, D.-Y. (1988). A Learning Model for the Selection of Problem Solving Strategies in Continuous Physical Systems. In *Proceedings of the Fourth IEEE Conference on Artificial Intelligence Applications*, pages 200–206, San Diego, CA.

Zukier, H. (1986). The Paradigmatic and Narrative Modes in Goal-Guided Inference. In R. Sorrentino and E. T. Higgins, editors, *Handbook of Motivation and Cognition: Foundations of Social Behavior*, Guilford Press, New York, NY.

In Michalski, R. S., and Wnek, J. eds., 1996. *Proceedings of the Third International Workshop on Multistrategy Learning*, 135-146 Menlo Park, CA: AAAI Press.

An Empirical Study of Computational Introspection: Evaluating Introspective Multistrategy Learning in the Meta-AQUA System

Michael T. Cox
Computer Science Department
Carnegie Mellon University
Pittsburgh, PA 15213-3891
mcox+@cs.cmu.edu

Abstract

The theory of introspective multistrategy learning proposes that three transformations must occur to learn effectively from a performance failure in an intelligent system: Blame assignment, deciding what to learn, and learning-strategy construction. The Meta-AQUA system is a multistrategy learner that operates in the domain of story-understanding failures and is designed to evaluate this learning approach. We discuss experimental results supporting the hypothesis that introspection facilitates learning in a multistrategy environment. In an empirical study, Meta-AQUA performed significantly better with a fully introspective mode than with a reflexive mode in which learning goals were ablated. In particular, the results lead to the conclusion that the process which posts learning goals (deciding what to learn) is a necessary transformation if negative interactions between learning methods are to be avoided and if learning is to remain effective. Moreover, we show that because learning algorithms can negatively interact, the arbitrary ordering of learning methods can actually lead to worse system performance than no learning at all. The goals of this research are to better understand the interactions between learning algorithms, to determine the role of introspective mechanisms when integrating them, and to more firmly establish the conditions under which such an approach is warranted (and those under which it is not).

Introduction

From the very early days of AI, researchers have been concerned with the issues of machine self-knowledge and introspective capabilities (e.g., McCarthy 1959; Minsky 1954/1965), yet few have quantitatively evaluated the trade-offs involved with much care or investigated the nature of the role introspection assumes when learning. The research presented here uses computational introspection to assist in the choice and sequencing of learning algorithms within a multistrategy framework. Yet open questions exist as to whether introspection is worth the computational overhead and in exactly what ways it facilitates the learning process. This paper begins to investigate these research questions empirically.

The theory of *introspective multistrategy learning* (IML) proposes that three transformations must occur to learn effectively from performance failures. First, given a trace of the performance failure, a learner must perform *blame assignment* by mapping the symptoms of the failure to a causal explanation of the failure. Secondly, the learner must use this explanation to *decide what to learn* by posting explicit learning goals to achieve desired changes in its background knowledge. Thirdly, the learner can use these goals for *learning-strategy construction* by treating the learning task as a nonlinear planning problem. That is, the learner constructs a partially-ordered plan to repair the background knowledge by sequencing calls to standard learning algorithms. The Meta-AQUA system (Cox 1996; Ram & Cox 1994) is a multistrategy learner that operates in the domain of story understanding failures and is designed to evaluate this learning approach.

Section 2 briefly presents the Meta-AQUA system by describing the story generation module with which experimental trials are generated and by providing a brief explanation of the performance and learning tasks. Section 3 provides a computational evaluation of the hypothesis that introspection facilitates learning using data obtained from the Meta-AQUA system. Section 4 summarizes the results and concludes with a short discussion.

Meta-AQUA

Meta-AQUA is a learning system that chooses and combines multiple learning methods from a toolbox of algorithms in order to repair faulty components responsible for failures encountered during the system's performance task. The system architecture and flow of information within Meta-AQUA is shown in Figure 1. The problem generation module outputs a story to the story-understanding performance system with the initial goal to understand the input. The performance module uses schemas from its background knowledge (BK) to explain the story and to build a representation for it in its foreground knowledge (FK). If this task fails, then a trace of the reasoning that preceded the failure is passed to the learning subsystem. A case-based reasoning (CBR) (Kolodner, 1993) subsystem within the learner uses past cases of introspective reasoning from the BK to explain the comprehension failure and to generate a set of learning goals. These goals, along with the trace, are then passed to a nonlinear planner. The planner subsequently builds a learning strategy from its toolbox of learning methods. The learning plan is passed to an execution system that examines and changes items in the BK. These changes enable improved future performance.

The above conceptualization of learning is consistent with both Michalski's (1994) Inferential Learning Theory that decomposes a learning task into an input, the BK, and a learning goal and Carbonell (1986) and Veloso's (1992) emphasis on reasoning from a trace of the derivation of a solution rather than from solutions themselves. Although the algorithms and knowledge structures used by Meta-AQUA have been reported elsewhere (e.g., Cox 1994, 1996; Cox & Ram 1995; Ram & Cox 1994; Ram, Cox, & Narayanan 1995), this section outlines the system in order to provide a context for understanding the evaluation.

The Input: Elvis World and Tale-Spin

To support large data collection, the Tale-Spin story generation program[1] provides a potentially infinite number of input variations that test Meta-AQUA's ability to learn from explanation failure. Given a main character and a problem, Tale-Spin simulates the actions that would be necessary for the character to achieve goals stemming from the problem. For example if a character is bored, Tale-Spin assigns the character an initial goal to remove the state of boredom. The character can achieve the goal by convincing a friend to play, finding a ball, going outside, and then batting the ball back and forth (see Figure 2). For each event in the story, the generator adds any associated causal results. These results change the world and enable further actions by characters in

[1]Tale-Spin (Meehan 1981) was obtained from the UC Irvine repository. Pazzani (1994) used it to evaluate the OCCAM multistrategy learning system.

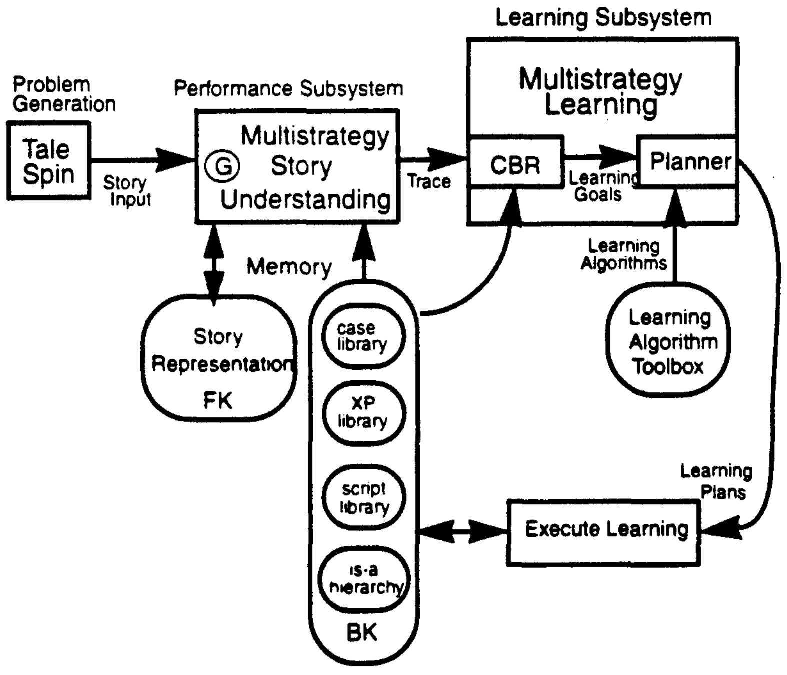


Figure 1. Detailed Meta-AQUA system architecture

the story. For example, the act of getting the ball and going outside enables the hitting of the ball which results in the ball's movement between the characters. In turn, these actions remove the boredom. The story terminates when the goals and subgoals of the main character have been achieved or when all possible plans to achieve them have been exhausted.

```
Elvis asked Lynn, "Would you push the
ball2 to me away from you?" Lynn went to
the garage. She picked up the ball2. She
had the ball2. She went to outside. He
went to outside. He played with the
ball2. She hit the ball2. She hit the
ball2 because she wanted to move the
ball2 to him. He hit the ball2. He hit the
ball2 because he wanted to move the ball2
to her. He played with the ball2 because
he didn't want to be bored.
                --- The End ---
```

Figure 2. Sample Elvis World story

Among the changes to Tale-Spin, we added a musician named Elvis and a police officer to the cast of characters. Elvis is temporarily boarding with Mom, Dad and their daughter Lynn, whereas the officer occasionally visits the house, presumably because of neighborhood complaints of loud music and raucous behavior. Furthermore, the police officer often (but not always) brings a drug-detection dog along with him. We also removed Karen from the cast of main characters available as a protagonist and the state of hunger from the possible initial problem states. Thus, Tale-Spin now generates a more uniform distribution of situations.

We also added two new problem types to the original problems of thirst and boredom. Characters may now be *jonesing*[2] for drugs. In Elvis' case, he sometimes smokes marijuana to relieve his jones, whereas Dad occasionally smokes a pipe with tobacco. Lynn has also been given a tobacco habit. The police officer has the problem of being *concerned* about the law. The police officer's state of being concerned is relieved if he can either locate contraband or arrest criminals.[3] We also reprogrammed Tale-Spin to hide the marijuana during story initialization in different locations (e.g, in the cupboard, refrigerator, and under the carpet), so the officer's task varies depending on entry conditions (i.e., at what point in the story the officer arrives on the scene and whether the dog accompanies him), the initial location of the pot, and the actions of the characters in the story.

Finally, to facilitate the performance task, we modified the Tale-Spin program so it generates explanations of key events in the stories. The resolution of all anomalies are thereby incorporated within each story. For example, Tale-Spin includes a reason why Lynn strikes the ball in the story above because it knows that Meta-AQUA will find the action anomalous and thus try to explain it. Although in an ideal implementation, the understanding process should be able to make independent inferences that confirm explanations of the input, Meta-AQUA depends on the story to provide explanations for this confirmation. The implementation concentrates on the learning task rather than the understanding task.

The Performance and Learning Tasks: Story Understanding, Explanation, and Repair

The Meta-AQUA system learns about drug-smuggling and sports activities, given its prior experience with stories about terrorists and its general knowledge of physical causality. The systems' performance task is to "understand" stories by building causal explanations that link the individual events into a coherent whole. The performance sub-system uses a multistrategy approach to understanding. Thus, the top-level goal is to choose a comprehension method (e.g., script processing, case-based reasoning, or explanation pattern application) by which it can understand the input. When an anomalous or otherwise interesting input is detected, the system builds an explanation of the event, incorporating it into the preexisting model of the story in the FK.

[2]In the vernacular, a "jones" is a drug habit accompanied by withdrawal symptoms. The verb "to jones" is to be going through a state of withdrawal.

[3]Unlike the UC Irvine version of Tale-Spin in which characters and their goals did not interact, we modified the program so that the police officer is a competing character with his own problem and goal. Because the officer confiscates the marijuana when found and arrests Elvis, such events may preempt the enabling conditions of actions Elvis had planned to perform. For instance, if Elvis is thirsty but the officer arrests him, this condition restricts his freedom of movement so that he cannot go to the faucet for water. Therefore, the story can end with Elvis still having the problem with which he began (i.e. thirst).

In the story from Figure 2 for example, Meta-AQUA finds it unusual for a person to strike a ball because its conceptual definition of the "hit" predicate constrains the object attribute to animate objects. It tries to explain the action by hypothesizing that Lynn tried to hurt the ball (an abstract explanation pattern, or XP, retrieved from the BK instantiates this explanation). In the following, sentence, however, the story specifies an alternate explanation (i.e., the hit action is intended to move the ball to the opposing person). This input causes an expectation failure because the system had expected one explanation to be true, but another proved true instead.

When the Meta-AQUA system detects an explanation failure, the performance module passes a trace of the reasoning to the learning subsystem. At this time, the learner needs to explain why the failure occurred (assign blame) by applying an introspective explanation to the trace. A meta-explanation (Meta-XP) is retrieved using the failure symptom as a probe into memory. Meta-AQUA instantiates the retrieved meta-explanation and binds it to the trace of reasoning that preceded the failure. The resulting structure is then checked for applicability. If the Meta-XP does not apply correctly, then another probe is attempted. An accepted Meta-XP either provides a set of learning goals (determines what to learn) that are designed to modify the system's BK or generates additional questions to be posed about the failure. Once a set of learning goals are posted, they are passed to the nonlinear planner for building a learning plan (strategy construction).

Figure 3 lists the major state transitions that the three learning processes produce. The learning plan is fully ordered to avoid interactions. For example, the abstraction step must precede the other steps. A knowledge dependency exists between the changes on the hit concept as a result of the abstraction and the use of this concept by both generalization and the indexing.[4] After the learning is executed and control returns to sentence processing, subsequent sentences concerning the hit predicate causes no anomaly. Instead, Meta-AQUA predicts the proper explanation.[5]

Symptoms:
- Contradiction between input and background knowledge
- Contradiction between expected explanation and actual explanation

Faults:
- Incorrect domain knowledge
- Novel situation
- Erroneous association

Learning Goals:
- Reconcile input with conceptual definition
- Differentiate two explanations

Learning Plan:
- Abstraction on concept of hit
- Generalization on hit explanation
- Index new explanation
- Mutually re-index two explanations

Figure 3. Learning from Explanation Failure

Computational Evaluation

This section presents the results of computational studies performed with Meta-AQUA to test the hypothesis that introspection facilitates learning. The methodology below not only tests our hypothesis, but also it more directly supports the position that a loose coupling of blame-assignment and repair (via learning goals) is preferred to a tight coupling approach. But perhaps more importantly, this methodology also scrutinizes the claim that the second phase of learning, deciding what to learn, is *necessary* for effective learning. IML theory is the only learning theory that makes such a strong claim. Few computational systems other than Meta-AQUA include an explicit calculation of a goal to learn and then use that goal to influence learning. Converging with the arguments and hand-coded examples from previous research that favor this position (e.g., Cox 1994; Cox & Ram 1995),

[4]During mutual re-indexing, the explanations are differentiated based on the object attribute-value of the hit. However, the abstraction transmutation changes this attribute. The generalization method applied to the new explanation also uses this attribute. See Cox & Ram (1995) for a more complete analysis.

[5]As pointed out by an anonymous reviewer, it would be nice for the system to use ontological knowledge to infer that the inanimate objects cannot feel pain. At the current time, however, the system possess neither the bias to make a proper inductive leap during learning nor the prerequisite knowledge to make the inference. Indeed, the system has but a primitive causal understanding of the mechanics of pain.

this paper presents quantitative evidence that supports the utility of this stage.

The Hypothesis

Generally, we claim that introspection facilitates learning. More specifically, we assert that the rate of improvement in story understanding with learning goals exceeds that of story understanding without learning goals holding all other factors constant. Our approach is to perform a kind of ablation study. Surgically removing the learning goals eliminates part of the system's mechanism responsible for introspection. The intention of this manipulation is to show different empirical learning curves with and without introspection as a function of the number of inputs.

Introspective learning is a computational process with the decomposition as shown in the upper portion of Figure 4. *Fully introspective multistrategy learning* consists of examining one's own reasoning to explain where the reasoning fails. It consists further of knowing enough about the self and one's own knowledge that the reasoner can explicitly decide what needs to be learned. Introspection amounts to performing blame assignment and subsequently posting explicit goals to learn. Learning amounts to the construction of a learning plan designed to change the reasoner's knowledge and thereby to achieve the learning goals.

Removing the goals from the introspective process above, leaves a more reflexive activity we call *semi-introspective multistrategy learning*[6] (see the lower portion of Figure 4). Instead of using the explanation of failure created during the blame-assignment phase to post a set of learning goals that then direct the construction of a learning plan, the explanation can directly determine the choice of repair methods. System performance under both conditions can then be compared with Meta-AQUA under a no learning situation.

Independent and Dependent Variables

Learning rates relative to a baseline no-learning condition are compared between the fully introspective and a semi-introspective version of Meta-AQUA. The independent variable that effects this change is the presence and influence of learning goals. The first experimental condition is referred to as the learning goal (LG) condition, and represents Meta-AQUA as described in Ram & Cox (1994). Under the LG condition, the system builds a learning strategy. This construction is guided by the learning goals spawned by the Meta-XPs that explain the failure. Hence, this condition represents a loose coupling approach between fault (failure cause) and repair (learning).

The second condition is called the random learning (RL) condition. Given the explanation of the causes of failure the system can directly assign calls to particular learning algorithms for each fault. The construction of the learning plan is then performed by a random ordering of these function calls, rather than by non-linear planning to achieve the learning goals. The RL condition represents a tight coupling

[6]It is semi-introspective because, although part of the introspective process has been removed, the introspective mechanics of blame-assignment remain. Future research remains to test the performance with blame assignment removed and learning goals present.

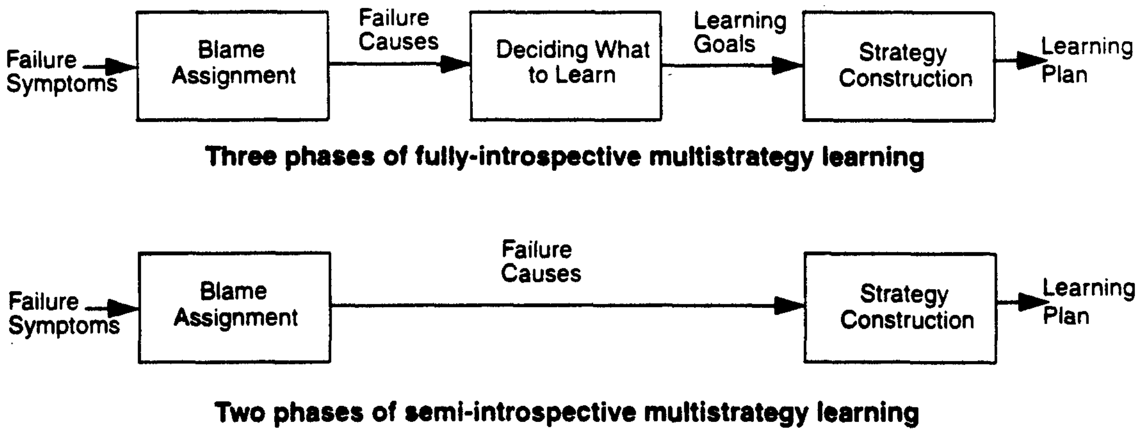


Figure 4. Learning goal ablation

approach (i.e., direct mapping from fault, or failure cause, to repair).

The final condition is called the no learning (NL) condition in which Meta-AQUA performs story understanding, but if a failure exists, the system constructs no learning strategy. This condition represents the baseline performance from which both the LG and RL conditions can be compared. Holding all parameters constant except the independent variables, Meta-AQUA is given input from the Tale-Spin problem generator and the dependent variable is measured.

The dependent variable must measure system performance (i.e., story understanding and explanation). In previous research, paraphrase and question answering tasks have been used as this measure (e.g., Lehnert, Dyer, Johnson, Yang, & Harley 1983; Schank & Riesbeck 1981; Wilensky 1978). If a reader sufficiently understands a body of text, the reader should be able to summarize the central points of the story and list the major events within it. If the story is well understood, then the reader can answer questions concerning the events and relationships within the story.

With story understanding programs such as BORIS (Lehnert et al. 1983), the researchers pose questions to the system and subjectively evaluate the answers to determine text comprehension effectiveness. One can count the number of questions answered correctly to ascertain an "absolute" measure of performance, but this is misleading. In contrast to externally posed questions, Chi (1995, Chi et al. 1989) reports that improved learning is correlated with human subjects who generate their own questions and explain the answers themselves. Being able to recognize that a gap exists in one's own knowledge, and thus to ask the question "Why don't I understand this?" (Ram 1991), is the first step to improving understanding. To pose self-generated questions thus indexes understanding and simultaneously reduces the probability of asking only easy questions. So, to evaluate the ability of the Meta-AQUA system, credit is given for simply posing a question that deserves asking.

Moreover, humans who are asked questions on reading tests are sometimes given points for partial answers. Unlike questions that have provably correct answers, answers to explanatory questions are difficult to judge in an absolute sense. So to be more realistic, the evaluation criterion in Meta-AQUA assigns credit for providing any answer to a question. Therefore, the full evaluation metric is as follows. For each anomalous or interesting input in a story, a point is given for posing a question, an additional point is given for providing any answer whatsoever, and a third point is assigned for answering what the researcher judges correct. The sum represents the dependent variable.

The Empirical Data

To serve as experimental trials and to minimize order effects, Tale-Spin generated six random sequences of Elvis-World stories. On each of these runs, Meta-AQUA processes a sequence three times, once for each experimental manipulation. The system begins all runs with the same initial conditions. For a given experimental condition, it processes all of the stories in the sequence while maintaining the learned knowledge between stories. At the end of the sequence, the system resets the BK. The input size for a run varies in length, but averages 27.67 stories per run.[7] The corpus for the six runs includes 166 stories, comprising a total of 4,884 sentences. The stories vary in size depending on the actions of the story and Tale-Spin's randomness parameters (e.g., the probability that a character will stop throwing an object on the current toss), but average 29.42 sentences.

Run Number Four. Run number four is particularly interesting because the greatest number of learning interactions occur in this set. The input to run four consists of 24 stories as enumerated in Table 1. The stories contain a total of 715 sentences, and the average number of sentences per story is 29.8. Each numeric entry in Table 1 contains a triple of the form <LG, RL, NL>. For example, the sixth column represents the number of learning episodes for each trial and for each condition. Note that the third element of each triple in this column is zero since learning is disabled in the NL condition. The fifth column (Question Points) contains the values for the dependent variable. These values represent the sums of triples from the second, third and fourth columns (Posed Questions, Answered Questions and Correct Answers, respectively).

In this run, random drug busts occur 11 times (5 with the canine squad and 6 with a lone police officer). Also, Dad is the most common protagonist, while Elvis, Officer1, and Lynn are tied for the least common. Furthermore, boredom is the major problem encountered by the main characters, although considering the number of random drug busts, the household can hardly be classified as sedate. The main characters solve (or attempted to solve) seven of these boredom problems by playing with one of three balls and solve three by playing with balloons. The state of being concerned is the least recurrent problem exhibited in the run.

[7]The reason that each run varies in length is that, after generating around 600,000 gensyms, Meta-AQUA will use all available swap space on the Symbolics and thus inadvertently halt the underlying LISP system. We then discard the story which is being processed at the time of the crash. The data from the remaining stories constitute the results of the run.

Table 1: Results from run number four

Story Number (sentences)[a]	Questions Posed (LG RL NL)	Answered Questions (LG RL NL)	Correct Answers (LG RL NL)	Question Points (LG RL NL)	Learning Episodes (LG RL NL)	Protagonist and Problem[b]
1 (26)	1 1 1	0 0 0	0 0 0	1 1 1	1 1 0	Mom bored (balloon)
2 (19)	3 3 3	3 2 2	1 0 0	7 5 5	2 3 0	Mom bored (ball)
3 (38B)	1 1 1	0 0 0	0 0 0	1 1 1	1 1 0	Elvis jonesing
4 (51b)	1 1 1	1 0 0	1 0 0	3 1 1	0 1 0	Dad jonesing
5 (21)	1 1 1	1 0 0	1 0 0	3 1 1	0 1 0	Mom bored (ball)
6 (13)	1 1 1	1 0 0	1 0 0	3 1 1	0 1 0	Officer1 concerned
7 (13)	1 1 1	1 0 0	1 0 0	3 1 1	0 1 0	Dad bored (ball)
8 (21)	0 0 0	0 0 0	0 0 0	0 0 0	0 0 0	Dad thirsty
9 (44B)	2 2 2	2 1 1	1 0 0	5 3 3	1 2 0	Dad thirsty
10 (51B)	3 3 3	2 1 1	2 1 0	7 5 4	0 1 0	Dad bored (balloon)
11 (11)	2 2 1	1 1 1	1 0 0	4 3 2	1 2 0	Lynn bored (ball)
12 (3)	0 0 0	0 0 0	0 0 0	0 0 0	0 0 0	Officer1 concerned
13 (47b)	2 2 1	1 1 0	1 1 0	4 4 1	0 0 0	Mom thirsty
14 (15)	4 4 4	4 2 3	4 0 0	12 6 7	0 4 0	Mom bored (ball)
15 (28)	0 0 0	0 0 0	0 0 0	0 0 0	0 0 0	Lynn jonesing
16 (42B)	2 2 2	2 1 1	2 1 0	6 4 3	0 1 0	Dad jonesing
17 (45b)	2 2 1	1 1 0	1 1 0	4 4 1	0 0 0	Elvis jonesing
18 (21)	2 2 2	2 1 1	2 1 0	6 4 3	0 1 0	Officer1 concerned
19 (20)	0 0 0	0 0 0	0 0 0	0 0 0	0 0 0	Dad jonesing
20 (52b)	2 2 1	1 0 0	1 0 0	4 2 1	0 1 0	Dad bored (balloon)
21 (39b)	2 2 1	1 1 1	1 1 1	4 4 3	1 1 0	Lynn jonesing
22 (17)	2 2 2	2 1 1	2 0 0	6 3 3	0 2 0	Dad bored (ball)
23 (40B)	2 2 2	1 1 1	1 1 0	4 4 3	1 1 0	Elvis thirsty
24 (38b)	2 2 1	1 1 0	1 0 0	4 3 1	0 1 0	Mom bored (ball)
Total 715	38 38 32	28 15 13	25 7 1	91 60 46	8 26 0	

a. The letter "B" means that the story contains an attempted drug bust by the police canine squad, whereas the letter "b" means that the officer entered the house alone to attempt a bust.

b. Items in parentheses represent games played to dispel boredom.

Table 2: Summary of results from run four

Learning Condition	Questions Posed	Answered Questions	Correct Answers	Total Question Points	Learning Episodes
LG	38	28	25	91	8
RL	38	15	7	60	26
NL	32	13	1	46	0

Table 2 summarizes the totals from Table 1.The dependent variable (column 5) shows that Meta-AQUA's performance under the LG condition is significantly greater than the performance under the RL condition. In turn, Meta-AQUA performance in the RL condition far exceeded the performance under the NL condition.

Alternatively, if only absolute performance (column 4) is considered, the differential is even greater. By this measure, the LG condition is more than three times the value of the RL condition, whereas, the performance of the NL condition is insignificant. By looking at column three, however, the numbers of questions answered in some way (right or wrong), are roughly equivalent in the RL and NL conditions, whereas the ratio of the LG condition to either of the other two is 2:1. Finally, the number of questions posed are virtually equal across all three conditions.

In contrast to these differences, Meta-AQUA attempts to learn from failure more than three times as often under the RL condition as under the LG condition. That is, learning is more *effective* with learning goals than without. In the RL condition, learning does not increase performance as much as does the LG condition, while concurrently, it leads Meta-AQUA to expend more resources by increasing the amount of learning episodes. Thus, the system works harder and gains less under RL than under LG.

Figure 5 graphs the accumulation of question points across trials (i.e., stories).[8] The behavior of the system as measured by the dependent variable is greatest under the LG condition, next best under RL, and worst under the NL condition. But, the trend does not hold for each trial. Figure 6 shows raw scores indicating that the NL condition actually outperforms the RL condition on trial number 14. The reason for this effect is that under worse-case conditions, if the interactions present between learning methods are negative, the performance may actually degrade. As a result, randomly ordered learning may be worse than no learning at all.

The differences as a function of the independent variable are even more pronounced if only accuracy (the number of correct answers) is examined and partial credit ignored. Figure 7 shows that under the RL condition, Meta-AQUA did not answer a question correctly until trial number 10, whereas under the NL condition, it did not perform correctly until trial 21. On the other hand, because under the LG condition the system learned a new explanation early in trial number 1, it was able to answer a question by trial number two. This striking result was facilitated by the random order of input (i.e., the second trial happened to be about the same problem as the first) as well as by computational introspection.

Overall Results. Table 3 summarizes the evaluation data from the six program runs. As is evident across all runs, the LG condition consistently outperforms the RL condition in the total cumulative question points. In turn, the RL condition outperforms the NL condition, despite the occasional poor performance due to negative interactions. As indicated by the standard deviations, the amount of differences between and within conditions exhibit high variability across runs.

Given these totals, the percent improvement for either learning condition over the NL base condition is simply the ratio of the difference in the base performance score and either score to the base score itself. Thus for run one, the ratio of the difference between the LG and NL conditions (35 points) to the NL condition (50 points) is .7, or 70 percent. Again, the improvement in performance for the LG condition is consistently higher than that of the RL condition. This difference is calculated in the final column. The differential is the percent improvement of the LG condition over the RL condition and is computed by the same measure as was the improvements in the individual learning conditions. That is, the differential is the ratio of the difference between the two improvements to the lower rate.[9] Thus, the differential between the LG rate of learning in run number one and that of the RL condition is the ratio of the difference (8 percentage points) to the RL percentage (62). Hence, the ratio is .129, or an improvement of nearly 13 percent

Although the average differential between the two learning conditions (i.e., between fully-introspective and semi-introspective multistrategy learning) is more than 106 percent with a large standard deviation, this figure still overstates the difference. The expected gain in learning is more conservative. The differential between the average LG improvement (102.70) and the average RL improvement (65.67) is a 56.38 percent difference. That is, across a number of input conditions, the use of learning goals to order and combine learning choices should show about 1.5 times the improvement in performance than will a straight mapping of faults to repairs when interactions are present.

[8]Note that the final extent of all three curves reach the value of the triple in the totals column for column five.

[9]Note that this ratio can also be calculated as the difference between the performance scores of the learning conditions to the difference between the performance score of the RL and NL conditions. In other words, the ratio (LG-RL) / (RL-NL).

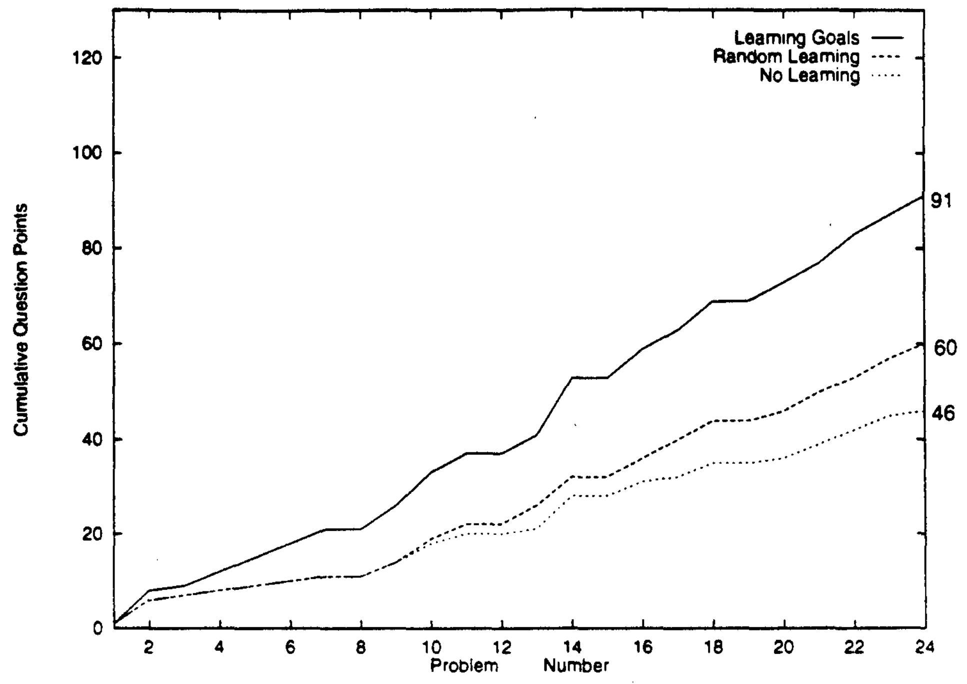


Figure 5. Run 4, cumulative question points as a function of the number of problems

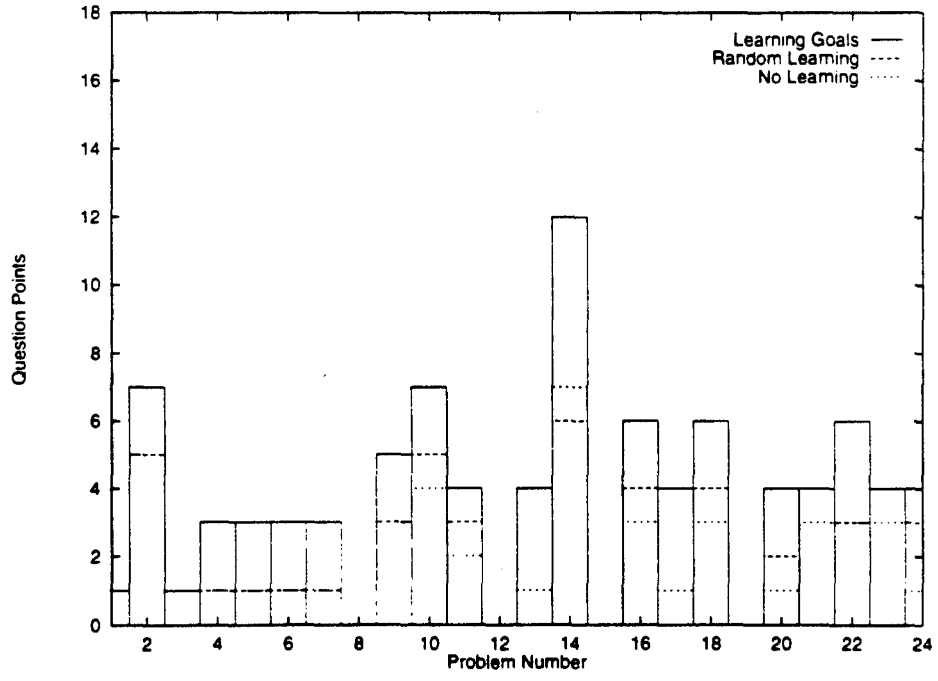


Figure 6. Run 4, question points histogram

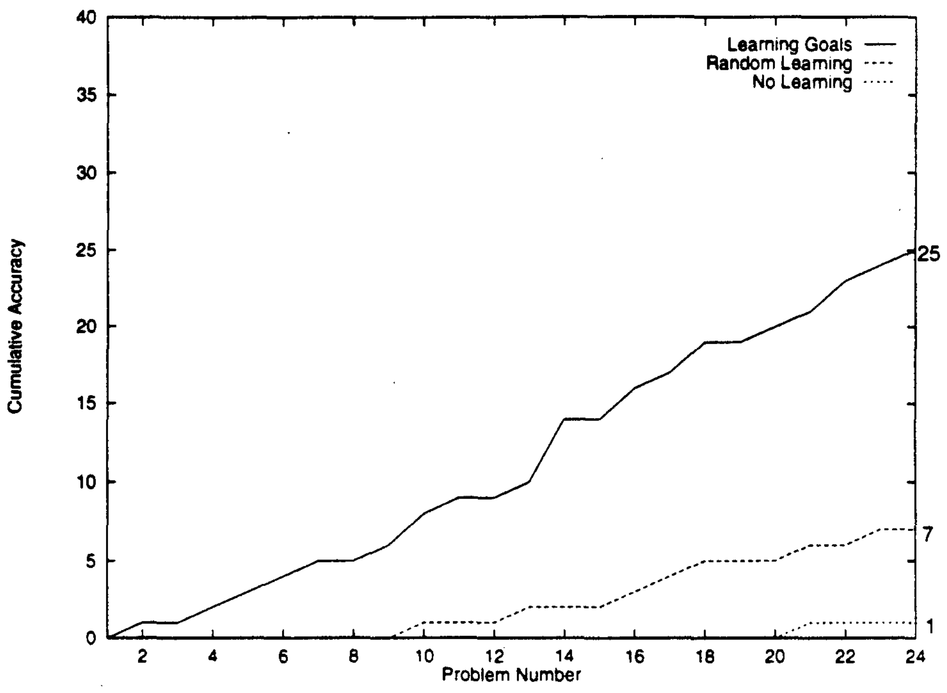


Figure 7. Run 4, cumulative correct answers (accuracy) as a function of the number of problems

Table 3: Summary of cumulative results

Run Number[a]	Cumulative Question Points LG	RL	NL	% LG Improved	% RL Improved	Improvement Differential %
Run 1 (34)	85	81	50	70.00	62.00	12.90
Run 2 (30)	106	98	43	146.51	127.91	14.55
Run 3 (28)	120	102	60	100.00	70.00	42.86
Run 4 (24)	91	60	46	97.83	30.43	221.43
Run 5 (22)	57	49	27	111.11	81.48	36.36
Run 6 (28)	103	66	54	90.74	22.22	308.33
Averages	93.67	76.00	46.67	102.70	65.67	106.07
Std. Dev.	21.72	21.31	11.34	25.43	38.17	126.59

a. Amounts in parentheses indicate total number of stories in each run.

Summary and Discussion

The experiments reported in this paper provide a number of results that support the hypothesis that computational introspection facilitates multistrategy learning. Meta-AQUA expended more learning resources and induced less performance improvement without learning goals than it did under a condition that included them. Moreover, we have shown that because learning algorithms negatively interact, the arbitrary ordering of learning methods (i.e., as under the RL condition) can actually lead to worse system performance than no learning at all. Therefore, an explicit phase to decide exactly what to learn (i.e., to spawn learning goals or an equivalent mechanism) is *necessary* to avoid these interactions and to maintain effective learning in multistrategy environments. The paper also provided a novel quantitative measure with which to evaluate the comprehension process. As dependent variable, this partial credit metric provides rewards for both posing questions autonomously and giving some type of answer, as well as for getting answers correct.

Because of the considerable computational overhead involved in maintaining a reasoning trace, performing blame-assignment, spawning learning goals, and constructing a plan with which to pursue such goals, the benefits of using introspection must be substantial to justify the costs.[10] Furthermore, under extremely complex situations or in informationally impoverished circumstances, deciding on an optimal learning goal is certainly intractable. In such situations, it may be more beneficial to proceed without further reasoning, rather than to attempt to understand the exact causes of the failure. Knowing when a learning task is worth pursuing is itself an important skill to master for an intelligent system. Identifying the most appropriate conditions for the use of an introspective approach is therefore a desirable research goal. To establish only that introspection facilitates learning and that the model of introspection has some quality of reasonableness is not satisfactory. Although further inquiry into these conditions is left for future research, a number of remarks can be made at this time.

If the distributions of the kinds of failures generated by the performance task change the nature of the differences in the learning curves generated in the experiments used to establish the hypothesis, then applicability conditions can be established that predict when the utility of introspection exceeds its cost. The space of applicability conditions for introspection is expected to emerge from the taxonomy of causal factors presented in Ram, Cox, and Narayanan (1995). It has already been shown through the existing implementation that introspection in certain circumstances is tractable. Thus, a lower bound is already available. It is clearly not possible to reason in any effective manner if all possible failures occur at once or given an overly-sparse BK. So an analysis of the interaction of the taxonomized causal factors should result in a set of complex failures that can be programmed into Tale-Spin in order to produce various distributions of errors. Meta-AQUA is expected to have difficulty learning from some of the failure combinations within these error distributions. As with the ablation study, measures with and without introspection provide the independent variable for the evaluation of learning. The results should itemize the conjunctions of failure from which it is impossible to recover and those for which a reflexive or tightly coupled approach is more suited.

In the interim, a potential heuristic for deciding when to use an introspective approach is to qualitatively ascertain whether or not interactions between learning mechanisms available to the learner exist. If they exist, then the approach should be applied, otherwise a more reflexive approach is licensed. In speculation, another potential heuristic for determining that introspection is a win is to use a threshold for the number of failure symptoms above which introspection will not be attempted. Through experimentation, this threshold number should be obtained empirically given a distribution of known problem types and a random selection of problems from the distribution. The identification of such heuristics will enable the practical use of introspective methods in systems that cannot afford to squander precious resources with intractable computation.

[10]One must be cautious, however, when summarily dismissing introspection due to computational overhead costs alone. Doyle (1980) warns that to disregard the introspective component and self-knowledge in order to save the computational overhead in space, time, and notation is discarding the very information necessary to avoid combinatorial explosions in search (p. 30).

Acknowledgments

This research is supported by AFOSR under contract #F49620-94-1-0092 and by the Georgia Institute of Technology. I thank Ashwin Ram, Janet Kolodner, Mimi Recker, Kurt Eiselt, Tony Simon, Kenny Moorman, Juan Carlos Santamaria, Tucker Balch, and the anonymous reviewers for comments on various drafts of this paper. I also wish to gratefully acknowledge Mark Devaney's assistance in reprogramming Tale-Spin.

References

Carbonell, J. G. 1986. Derivational Analogy: A Theory of Reconstructive Problem Solving and Expertise Acquisition. In R. Michalski, J. Carbonell and T. Mitchell, eds.. *Machine learning II: An Artificial Intelligence Approach*, 371-392. San Mateo, CA: Morgan Kaufmann Publishers.

Chi, M. T. H. 1995. Revising the Mental Model as One Learns. Plenary address to the Seventeenth Annual Conference of the Cognitive Science Society. Pittsburgh (July 23).

Chi, M. T. H., Bassok, M., Lewis, M., Reimann, P., and Glasser, R. 1989. Self-Explanations: How Students Study and Use Examples in Learning to Solve Problems. *Cognitive Science* 13:145-182.

Cox, M. T. 1994. Machines that Forget: Learning from Retrieval Failure of Mis-Indexed Explanations. In A. Ram and K. Eiselt, eds., Proceedings of the Sixteenth Annual Conference of the Cognitive Science Society, 225-230. Hillsdale, NJ: Lawrence Erlbaum Associates.

Cox, M. T. 1996. Introspective Multistrategy Learning: Constructing a Learning Strategy under Reasoning Failure. Ph.D. diss., Technical Report, GIT-CC-96-06. College of Computing, Georgia Institute of Technology, Atlanta. Available URL: ftp://ftp.cc.gatech.edu/pub/ai/ram/git-cc-96-06.html

Cox, M. T., and Ram, A. 1995. Interacting Learning-Goals: Treating Learning as a Planning Task. In J.-P. Haton, M. Keane and M. Manago, eds., *Advances in case-based reasoning*, 60-74. Berlin: Springer-Verlag.

Doyle, J. 1980. A Model for Deliberation, Action, and Introspection. Ph.D. diss., Technical Report, TR-58, Department of Computer Science, Massachusetts Institute of Technology, Cambridge.

Kolodner, J. L. 1993. *Case-Based Reasoning*. San Mateo, CA: Morgan Kaufmann Publishers.

Lehnert, W., Dyer, M. G., Johnson, P., Yang, C., and Harley, S. 1983. BORIS - An Experiment in In-Depth Understanding of Narratives. *Artificial Intelligence* 20(1): 15-62.

McCarthy, J. 1959. Programs with Common Sense. In Symposium Proceedings on Mechanisation of Thought Processes, vol. 1, 77-84. London: Her Majesty's Stationary Office.

Meehan, J. 1981. Talespin. In R. C. Schank and C. Riesbeck, eds., *Inside Computer Understanding: Five Programs Plus Miniatures*, 197-258. Hillsdale, NJ: Lawrence Erlbaum Associates.

Michalski, R. S. 1994. Inferential Theory of Learning: Developing Foundations for Multistrategy Learning. In R. Michalski and G. Tecuci, eds.. *Machine learning IV: A Multistrategy Approach*, 3-61. San Francisco: Morgan Kaufmann.

Minsky, M. L. 1965. Matter, Mind, and Models. In Proceedings of the International Federation of Information Processing Congress 1965, vol. 1, 45-49. (Original work from 1954)

Pazzani, M. 1994. Learning Causal Patterns: Making a Transition from Data-Driven to Theory-Driven Learning. In R. Michalski and G. Tecuci, eds., *Machine learning IV: A Multistrategy Approach*, 267-293. San Francisco: Morgan Kaufmann.

Ram, A. 1991. A Theory of Questions and Question Asking. *The Journal of the Learning Sciences* 1(3&4): 273-318.

Ram, A., and Cox, M. T. 1994. Introspective Reasoning Using Meta-Explanations for Multistrategy Learning. In R. S. Michalski and G. Tecuci, eds., *Machine learning IV: A Multistrategy Approach*, 349-377. San Francisco: Morgan Kaufmann.

Ram, A., Cox, M. T., and Narayanan, S. 1995. Goal-Driven Learning in Multistrategy Reasoning and Learning Systems. In A. Ram and D. Leake, eds., *Goal-Driven Learning*, 421-437. Cambridge, MA: MIT Press/Bradford Books.

Schank, R. C., and Riesbeck, C. eds. 1981. *Inside Computer Understanding: Five Programs Plus Miniatures*. Hillsdale, NJ: Lawrence Erlbaum Associates.

Veloso, M. 1994. *Planning and Learning by Analogical Reasoning*. Berlin: Springer-Verlag.

Wilensky, R. 1978. Understanding Goal-Based Stories. Ph.D. diss., Technical Report, 140, Department of Computer Science, Yale University, New Haven, CT.

A Domain-Independent Algorithm for Multi-Plan Adaptation and Merging in Least-Commitment Planners

Anthony G. Francis, Jr. and Ashwin Ram[*]
College of Computing
Georgia Institute of Technology
Atlanta, Georgia 30332-0280
{centaur, ashwin}@cc.gatech.edu

Abstract

Solving problems in many real-world domains requires integrating knowledge from several past experiences. What is needed is a method that allows the merging of arbitrary numbers of plans at any point during the adaptation process and the dynamic extraction of relevant case subparts. Our solution is the Multi-Plan Adaptor (MPA), a method for merging partial-order plans in the context of case-based least-commitment planning. MPA provides this ability by extracting an intermediate goal statement from a partial plan, clipping a stored plan to the intermediate goal statement, and then splicing the clipping into the original partial plan. Because the cost of retrieval can potentially outweigh the benefits of adaptation, developing heuristics for deciding when to retrieve is a challenging problem. Our current implementation of MPA in the multistrategy reasoning system Janus uses an asynchronous, resource-bounded memory module called MOORE that initially retrieves its current "best guess" but continues to monitor the adaptation system, spontaneously returning a better retrieval as soon as it is found.

1. Introduction

Taking advantage of past experiences is the foundation of case-based reasoning. When confronted with a problem, a case-based reasoner recalls a past experience and adapts it to provide the solution to the new problem. Unfortunately, in many real-world domains we cannot count on a single past experience to provide the outline of a solution to our problems. For example:

- A graduate student asked to present his first paper at an overseas conference must draw on previous, separate experiences in preparing talks for conferences within his country and preparing his passport and flight arrangements for vacations outside of his country.
- A host planning his first large dinner party must recall both the outline of a menu as served at family gatherings and his separate experiences at preparing individual dishes for himself.
- A home hobbyist attempting his first large piece of furniture must recall both past examples of that type of furniture to provide a design and experiences with acquiring, assembling and finishing individual components.

All of these problems have something in common: every *piece* of the solution can be constructed entirely out of the agent's past experience (with suitable adaptation), but no *single* past experience suffices to solve the entire problem. For these types of problems, unless a case-based reasoning system has the ability to combine several past experiences, it will have to resort to expensive from-scratch reasoning in order to solve the problem.

Some CBR planning systems combine multiple cases during reasoning. However, they either gather all partial plans at retrieval prior to adaptation (e.g., PRODIGY/ANALOGY, Veloso 1994), or break plans into *snippets* at storage time so they can be retrieved individually (e.g., CELIA, Redmond 1990, 1992). Neither of these approaches are entirely satisfactory, for various reasons.

It is not entirely clear that all of the knowledge needed to solve a problem can be assembled at the beginning of problem solving. For example, in the furniture example, it is not entirely clear whether or not the agent needs to buy new sandpaper, and hence unclear whether the agent should recall past experiences of buying sandpaper at a hardware store. This uncertainty arises out of several concerns: the uncertainty of the world state (how much sandpaper does the agent have?), uncertainty in the effectiveness of agent actions (how much wood will a piece of sandpaper sand?) and the potential of exogenous events that can invalidate parts of the plan (if a friend drops and scars a piece of the furniture, will the agent have enough sandpaper to remove the scar, or will he need to buy more?). But it can also arise out of *the plan itself:* until the agent has decided on a design for the piece and how much wood will be involved, it is unclear precisely how much sandpaper is needed, and hence unclear whether or not a plan should be retrieved. If some amount of sandpaper is on hand, the goal of acquiring sandpaper may not even arise until late in the planning process, when it has become clear that the amount on hand is insufficient.

Precomputing case snippets also has drawbacks. While this allows us to extract subparts of a case to

[*] This research was supported by the United States Air Force Laboratory Graduate Fellowship Program, by the Air Force Office of Scientific Research under Contract # F49620-94-1-0092, and by the Georgia Institute of Technology.

meet the needs of a component of a plan, these subcomponents need to be computed at storage time. Unfortunately, it is not clear that every useful breakdown of a case can be computed in advance. For example, in the foreign conference example, the two past experiences must be closely interleaved to produce a new plan. If the agent has not stored the passport experience as a separate snippet, the agent may not be able to extract that particular piece of a case if it is retrieved — if the agent was able to retrieve the case at all.

What is needed is a method that allows the merging of arbitrary numbers of plans at any point during the adaptation process, and which allows the dynamic extraction of relevant case subparts. Our solution is the Multi-Plan Adaptor (MPA), a novel method for merging partial-order plans in the context of case-based least-commitment planning. The MPA algorithm builds on the Systematic Plan Adaptor (SPA; Hanks and Weld 1994), a case-based least-commitment planner that annotates partially ordered plans with dependency structures to allow later refitting and adaptation. SPA shows significant improvements over generative planning, but can only adapt one plan at a time and hence for the types of problems described above must resort to significant amounts of from-scratch planning.

MPA overcomes this limitation of SPA by extracting an *intermediate goal statement* from a partial plan, *clipping* a stored plan to the intermediate goal statement, and then *splicing* the clipping into the original partial plan. Depending on the size of the plans spliced and the retrieval algorithms used, MPA can produce significant speedups over SPA.

Because the cost of retrieval can potentially outweigh the benefits of adaptation in an interleaved system, developing heuristics for deciding when to retrieve is a challenging problem. Our current implementation of MPA in the multistrategy reasoning system Janus uses an asynchronous, resource-bounded memory module called MOORE that initially retrieves its current "best guess" but continues to monitor the adaptation system, spontaneously returning a better retrieval as soon as it is found.

In this paper, we briefly review least-commitment planning and how it can be used for adaptation. We then present the MPA algorithm, describe its foundations in and extensions of the SPA algorithm, and then detail how MPA can be used to address the limitations of existing multi-plan systems. We then discuss how MPA can be integrated into various control regimes, including systematic, pure case-based and interleaved systems, and describe our implementation of interleaved MPA in the Janus system. We conclude the paper by reviewing other case-based planning work and then outline our contributions in the appendix.

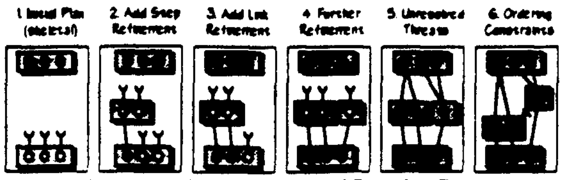


Figure 1. Refinement of Partial Plans

2. Least Commitment Planning and Systematic Plan Adaptation

2.1. Least Commitment Planning: SNLP

Least-commitment planning departs from traditional planning systems by delaying decisions about step orderings and bindings as much as possible to prevent backtracking (Weld 1994). Depending on the domain and search strategy, least-commitment planning can lead to substantial improvements over traditional totally ordered plans (Barret & Weld 1994; but for an alternative view see Minton et al 1994, Veloso & Blythe 1994).

Least-commitment planners solve problems by successive *refinement* of a partial plan derived from the initial and goal conditions of the problem (Figure 1). Plans are represented as sets of steps, causal links between steps, variable bindings and ordering constraints. Beginning with a skeletal partial plan based on the initial and goal conditions of the problem, a least-commitment planner attempts to refine the plan by adding steps, links and constraints that eliminate open conditions or resolve threats.

An *open condition* in a partially ordered plan occurs when a plan step has a precondition that has no causal link to an effect in a prior step in the plan that establishes that condition. Open conditions can be resolved by adding a new step that establishes the desired effect and linking the precondition to it, or by linking the precondition directly to an effect already in the plan if one exists.

A *threat* in a partially ordered plan occurs when the condition established by the producing step in a causal link may be clobbered by the effects of another step before it is used by the consuming step in the link. Threats may be resolved by adding *ordering constraints* that move the threatening step before or after the steps in the causal link, or by adding *binding constraints* that ensure that the effects of the other step cannot unify with the step of the causal link.

Given an arbitrary sequence of decisions to add steps and bindings, there is no guarantee that the partial plan produced can be successfully refined into a correct solution. An *analytical failure* occurs when an incomplete partial plan cannot be further refined by adding new steps, links or constraints. In a purely generative planner, analytical failures can be resolved by backtracking to a point in the search space where the inconsistent choices have not been made and then selecting different choices.

SNLP (McAllester & Rosenblitt 1991) is a complete, consistent and systematic partial order planner that uses a propositional STRIPS notation to represent steps in its partial plans. Problem solving in SNLP begins with a list of initial conditions and goal conditions, which SNLP transforms into a skeletal plan with a dummy initial step whose postconditions establish the initial conditions and a dummy final step whose preconditions match the goal conditions. As outlined above, SNLP operates by repeated refinement in which open conditions are resolved and threats are eliminated.

2.2. Systematic Plan Adaptation: SPA

The Systematic Plan Adaptor algorithm (Hanks & Weld 1994) is an algorithm for case-based planning that incorporates SNLP into its adaptation mechanism. SPA provides vast speedups over SNLP's performance by retrieving single plans which it fits and adapts, yet it maintains SNLP's properties of soundness, completeness, and systematicity.

SPA is based on three key ideas:

- annotate partial plans with *reasons* for decisions
- add a *retraction mechanism* to remove decisions
- add a *fitting mechanism* to fit previous plans to current situations

SPA extends the SNLP algorithm by adding *reason* annotations to refinements added to the plan. Reason data structures take a middle ground between a derivational trace of a planner's activity (e.g., Ram & Cox 1994, Ihrig & Kambhampati 1994a, Veloso 1994) and an unannotated plan. Like the partial-order plan representation itself, which records necessary orderings and interactions between steps without specifying a precise ordering, reasons record the necessary dependencies between refinements without specifying the precise ordering in which the planner made the choices.

The reason data structures are first used to *fit* a retrieved plan to the new situation (Figure 2). The initial and goal conditions of a prior plan may not match the current situation, and it may contain steps and links that are not relevant to solving the new problem. To remedy this situation, SPA adjusts the initial and final conditions of the retrieved plan to match the current problem, and then uses the reason data structures to recursively eliminate steps that attempt to resolve deleted goal conditions and links that depend on deleted initial conditions.

The reason structures are also used to select refinements for *retraction*. Once a plan has been retrieved, it may contain steps and constraints that would lead to analytical failures in the current situation. Since SPA retrieves a plan and modifies it to fit the current situation, it does not have a past reasoning history to backtrack over to eliminate analytical failures. Instead, SPA considers not only adding new refinements to the partial plans it is considering but also retracting those refinements. The reason data structures are used to select a plan decision upon which no other decisions depend. That decision is then removed and all of the *alternative* decisions that could have been made are placed in the queue of partial plans the system is working on.

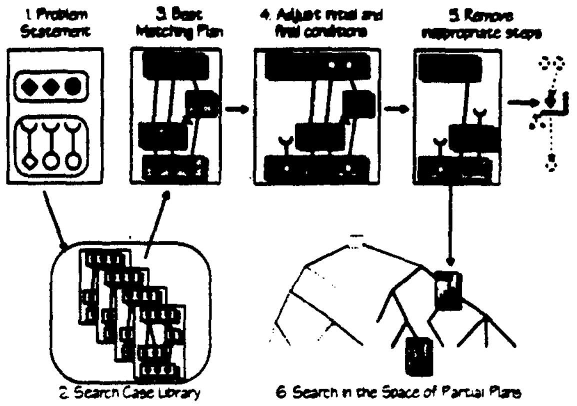

1 Problem Statement
3 Best Matching Plan
4 Adjust initial and final conditions
5 Remove inappropriate steps
2 Search Case Library
6 Search in the Space of Partial Plans

Figure 2. Fitting of Partial Plans

2.3. Limits of SPA: Achieving Systematicity

One limitation of SPA is that it can only adapt a single plan at a time: even if a new problem could be solved by retrieving and merging two plans, the system is constrained to retrieve only a single plan and then adapt it to fit. This limitation arises from SPA's attempt to maintain *systematicity*. Systematicity ensures that the problem solver never repeats work: if SPA considers a plan once, it never considers it again.

SPA ensures this by performing a breadth-first search in the space of partial plans, beginning with the retrieved prior plan. Initially, the set of states on the system's search frontier consists of only this plan, which is considered for both refinement and retraction. As search progresses, successive refinements are guaranteed (because of the properties of the underlying SNLP algorithm) never to backtrack over previously visited states. The system maintains systematicity in retraction by removing only one decision at a time. The retracted plan is added to the search frontier with an indication for further retraction, and all of the *alternative* decisions that the system might have made are added to the frontier for further refinement. Hence, only one plan on the search frontier is considered for retraction at any one time, and the result of a retraction produces a set of plans which are guaranteed to be new and guaranteed to never generate plans that have been previously considered.

3. An Algorithm for Plan Adaptation and Merging

SPA, while it does provide substantial speedups over SNLP, still is constrained to retrieving and applying a single plan, and therefore often must result to significant amounts of from-scratch planning even when the knowledge needed to complete the plan is present in the case library. To make the most effective use of the planner's past experience, we need the ability to recognize when a partial plan needs to be extended, select plans to that address the deficiency, and then extract and merge the relevant parts of the retrieved plan into the original partial plan.

The Multi-Plan Adaptor algorithm (MPA) resolves this problem in SPA by allowing the retrieval and merging of arbitrary numbers of cases at any point during the adaptation process. MPA further allows the dynamic extraction of relevant parts of past cases. To achieve this, the MPA algorithm extends the SPA framework in three crucial ways:

- extract intermediate goal statements from partial plans
- use the fitting mechanism to clip plans for merging
- build a plan splicer to merge two plans

Intermediate goal statements provide MPA with the ability to merge partial plans at any point of the adaptation process and contributes to its ability to dynamically extract the relevant subparts of retrieved cases. MPA translates the incomplete state of a partial plan into a goal statement, allowing the system to use the same retrieval mechanisms that it used to retrieve the initial plan. Once a plan has been retrieved that matches the intermediate goal statement, the relevant subparts must be extracted. The plan fitting mechanism performs this extraction dynamically, removing portions of the plan that are not causally relevant to the intermediate goal statement.

Intermediate goal statements are extracted by the inverse of the representational trick with which SNLP constructs its initial skeletal plans. Recall that SNLP builds the initial plan it considers by adding dummy initial and final steps whose post- and pre-conditions match the initial and goal conditions of the problem. As planning proceeds, open conditions in the goal statement are resolved, but new open conditions are posted as new steps are added. These open conditions themselves can be extracted from the plan to form a new goal statement. Similarly, the initial conditions of the partial plan can be extracted and used as a new set of initial conditions.[1]

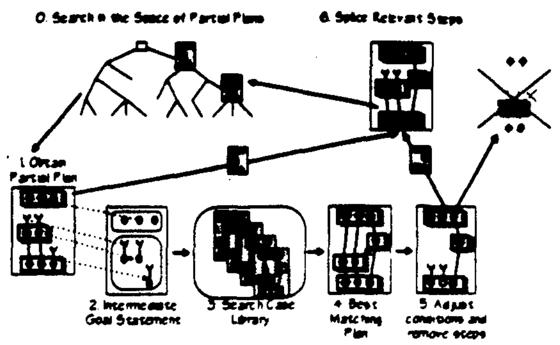

0. Search in the Space of Partial Plans
6. Splice Relevant Steps
1. Obtain Partial Plan
2. Intermediate Goal Statement
3. Search Case Library
4. Best Matching Plan
5. Adjust conditions and remove steps

Figure 3. Multi-Plan Adaptation

Just like the original goal statement, the intermediate goal statement can be used to retrieve and fit a partial plan. However, the result of this process is not a complete fitted plan suitable for adaptation; it is a *plan clipping* that satisfies some or all of the open conditions of the partial plan from which the intermediate goal statement was derived.

To take advantage of the plan clipping for adaptation, it must be *spliced* into the original partial plan (Figure 3). Our splicing mechanism uses the intermediate goal statement to produce a mapping between the partial plan and the plan clipping, pairing open conditions from the partial plan with satisfied goal conditions from the plan clipping. The plan splicer uses this mapping to perform a guided refinement of the original partial plan, selecting goal conditions from the clipping and using the links and steps that satisfied them as templates to instantiate similar steps and links in the original plan. As these steps are added, new mappings are established between open conditions in the new steps and satisfied preconditions in the clipping and are added to the queue of mappings that the splicer is processing. Hence, the plan splicer performs a backwards breadth-first search through the causal structure of the plan clipping, using links and steps in the clipping to guide the instantiation of links and steps in the original plan.

3.1. The Efficiency of Plan Splicing

Both adapting a single partial plan and adapting merged partial plans can produce significant benefits over generative problem solving. The cost of generative planning is exponential in the size of the final plan produced, whereas fitting a plan is a linear operation in the size of the plan. Hence, the potential exists for substantial improvement through retrieval and adaptation if an appropriate past plan exists, especially for large plans. In certain domains, SPA has demonstrated significant improvements over generative planning. However, if large gaps exist in the retrieved partial plan, SPA must resort to

[1] Unfortunately, since ordering constraints and binding constraints may be posted to the plan at any time, only the initial conditions can be guaranteed to be valid conditions for the intermediate goal statement. Conditions established by other steps of the plan may be clobbered by the addition of new steps and new ordering constraints. However, it might be possible to develop heuristics that select additional initial conditions that are likely to hold, perhaps in conjunction with more complex retrieval, fitting and splicing algorithms.

adaptation, which, like generative problem solving, has an exponential cost in the number of steps that need to be added.

While this amount of adaptation may be significant improvement over complete from-scratch problem solving, the potential exists to reduce that even further by using MPA to clip and splice more partial plans. Fitting a clipping and splicing a clipping are linear operations in the size of the plan being spliced. Hence, the potential exists for substantial improvement through plan merging if an appropriate past plan exists, especially if the gaps in the existing plan are large. An initial implementation of MPA for a test domain indicated significant speedups (beginning at 30%) over SPA for even the smallest examples (solution size of the final plan = 5 steps).

However, both plan adaptation and plan merging require the retrieval of a past plan, and that retrieval cost may offset the benefits of adaptation or merging. For SPA, retrieval costs are incurred once, before adaptation begins, and hence it is simple to determine whether the cost of retrieval will be offset by the benefits of adaptation. For MPA, the picture is not so simple. MPA allows plan merging at any point during adaptation, and hence it is not clear from the MPA algorithm itself when retrieval will be performed, how many retrievals will be performed, and hence whether those costs will be offset by the improvements in problem solving. In order to determine this, we must embed MPA within a *control regime* that determines when and how often retrieval will be performed before we can precisely specify the benefits of multi-plan adaptation.

4. Controlling Plan Splicing

Merely having the ability to splice plans together does not allow us to take advantage of past experience. We need to decide what experiences to combine and when to combine them. Because the MPA algorithm can potentially be performed at any point during the adaptation process — using an initial skeletal plan derived from the initial and goal statement, using a fitted plan derived from retrieval, or using an adapted plan after some arbitrary amount of retraction and refinement — we have considerable flexibility in deciding what to retrieve, when to retrieve it and when to merge it.

We have considered three alternative control regimes, each of which makes different commitments about when to retrieve and when to adapt. On one end of the spectrum, *Systematic MPA* preserves SPA's property of systematicity by splicing all retrieved cases before adaptation begins. On the other, *Extreme MPA* never performs (generative) adaptation and instead uses set of *pivotal cases* (Smyth & Keane 1995) to guarantee completeness.

Both Systematic and Extreme MPA make extreme commitments: either integrate all knowledge before adaptation begins, or never adapt and rely solely on past experience. An alternative approach is to allow plan splicing at any point during adaptation. In the middle stands *Interactive MPA* (MPA-i), a system that can potentially attempt a retrieval at any time, either with the initial skeletal plan or with partial plans produced as a result of adaptation. Since the results of splicing cause large jumps in the search space, the system deliberately departs from the systematicity of SNLP and SPA in an attempt to solve the problem with less search.

However, allowing arbitrary plan retrieval and plan splicing is not without cost. Performing a full search of the system's case library at every step of the problem space could be computationally prohibitive. The costs of searching the case library at every step of the problem space could outweigh the benefits of reduced search, especially if the system enters a "slump" — an adaptation episode which begins and ends with the application of relevant clippings, but which goes through a series of intermediate plans for which the system cannot match any existing plans in its case library. Clearly, it is worthwhile to retrieve and apply clippings at the beginning and end of a slump, but a full search of the case library at each intermediate step could cost more than the benefits that the initial and final retrievals provide. This is the *swamping utility problem* — the benefits of case retrieval can be outweighed by the costs of that retrieval, leading to an overall degradation in performance as a result of case learning (Francis & Ram 1995).

Developing heuristics for deciding when and when not to retrieve is a challenging open problem. To solve this problem, we have implemented a multistrategy reasoning system called Janus which pairs MPA with an *asynchronous*, resource-bounded memory module called MOORE. In Janus, MPA and MOORE share a central blackboard. When MPA requests a partial plan, MOORE returns its current best guess. However, the retrieval request remains active, and as the MPA adapts the plan the new partial plans it constructs provide additional cues for MOORE to attempt further retrievals. When MOORE finds a new past case whose degree of match exceeds a certain threshold, it signals MPA, which splices it into the appropriate partial plan.

5. Related Work

There are wide bodies of work on both least-commitment planning and case-based reasoning. The most relevant example of that work to this research is of course SPA, upon which MPA builds. MPA's plan splicing mechanism is in many ways similar to DERSNLP (Ihrig & Kambhampati 1994) derivational analogy system built on top of SNLP that uses *eager replay* to guide a partial order planner. While DERSNLP's eager replay mechanism is in some ways similar to a limiting case of MPA-s in which a single plan is retrieved and spliced into a skeletal plan derived from an initial problem statement, DERSNLP goes beyond SPA's reason mechanism and includes a full derivational trace of problem solving in its cases. While DERSNLP and its extension DERSNLP-EBL focus on when it is profitable to retrieve a partial plan, unlike MPA-i they do not provide the capability of interrupting adaptation as a result of an asynchronous memory retrieval, nor do they provide the ability to integrate the results of multiple plans.

Combining multiple plans in case-based reasoning is not a new idea. The PRODIGY/ANALOGY system (Veloso 1994) can retrieve and merge the results of an arbitrary number of totally ordered plans during the derivational analogy process. However, because PRODIGY/ANALOGY manipulates and stores totally ordered plans, it runs into significant issues on deciding how to interleave steps (Veloso 1994, p124-127), an issue MPA avoids because of its least-commitment heritage. Furthermore, all of the plans that the system merges must be retrieved prior to the beginning of adaptation; PRODIGY/ANALOGY does not have the ability to retrieve and combine cases on the fly.

The JULIA system (Hinrichs 1992) also has the ability to combine pieces of several past cases, but this is largely a domain-dependent algorithm for merging declarative structures, rather than a domain independent planning system. The CELIA system (Redmond 1990, 1992) stores cases as separate *snippets*, case subcomponents organized around a single goal or set of conjunctive goals. Snippets provide CELIA with the ability to retrieve and identify relevant subparts of a past case based on the system's current goals. Note that while snippets are superficially similar to plan clippings, plan clippings are constructed dynamically during problem solving, whereas snippets need to be computed and stored in advance.

Clippings are similar to macro operators (Fikes et al. 1972) in that they use past experience to combine several problem solving steps into a single structure that can be applied as a unit, allowing the system to make large jumps in the problem space and avoid unnecessary search. However, macro operators differ from clippings in two important ways. First, macro operators are precomputed at storage time, whereas clippings are computed dynamically; second, macro operators are fixed sequences of operators, whereas clippings are partially ordered sets of operators that may be resolved in a wide variety of ways in the final plan.

6. Conclusion

We have presented the Multi-Plan Adaptor, an algorithm that allows a case-based least-commitment planner to take advantage of the benefits of several past experiences. MPA provides the ability to retrieve and merge dynamically selected case components at any point during the adaptation process by extracting an intermediate goal statements from a partial plan, using the intermediate goal statement to retrieve and clip a past plan to the partial plan, and then splicing the clipping into the original partial plan.

Multi-plan adaptation has the potential for substantial speedup over single-plan adaptation, but in order for those benefits to be realized MPA must be embedded within a control regime that decides when the system attempts a retrieval, when the system merges, and when the system resorts to adaptation. We have implemented an interactive algorithm called MPA-i which allows the retrieval of past cases at any point during adaptation.

Because an interleaved system can potentially attempt a retrieval at *every* adaptation step and hence has the potential to swamp the benefits of adaptation under a sea of retrieval costs, developing heuristics for deciding when to retrieve is a difficult problem. Our current implementation of MPA-i as a part of the Janus multistrategy reasoning system uses an asynchronous, resource-bounded memory module called MOORE that retrieves a "best guess" and then continues to monitor the progress of adaptation, returning a new or better retrieval as soon as it is found.

Appendix: Describing the Contributions

A. Reasoning Framework:

1. *What is the reasoning framework?*
   The reasoning framework is a case-based reasoning system layered on top of a least-commitment planner. This framework can be embedded in several control regimes; an interleaved regime based on an asynchronous memory module is implemented.
2. *What benefits can be gained that motivated using adaptation? reuse?*
   Adaptation/reuse of past problem solving cases can eliminate vast amounts of search and cause significant speedups in problem solving.
3. *What are the specific benefits and limitations of your approach?*
   Our approach can provide improvement over from-scratch problem solving and single-case adaptation. However, it is currently limited to merging problem-solving cases or other cases with a causal structure, and like all planning and learning algorithms it is sensitive to domain characteristics.
4. *What invariants does it guarantee?*
   When the domain affords the use of past problem solving experiences and relevant experiences are available, this method can reduce the number of states visited in the search space.
5. *What are the roles of adaptation, knowledge and reuse in your approach?*

   A. Knowledge:
   1. *What does it encode?* Plans
   2. *How is it represented?* Partial-order plans with a propositional STRIPS notation.
   3. *How is it used?* To provide the framework for new plans.
   4. *How is it acquired?* Through generative or case-based problem solving.

B. Adaptation:

1. *What is adapted?* Plans (past problem solving cases).
2. *Why is it adapted?* To avoid generative search.
3. *What properties of the knowledge representation does adaptation require or exploit?* Plans must be annotated with reason data structures that allow retraction of decisions (for adaptation) and plan fitting (for adaptation and merging).
4. *How are the benefits measured?* In the number of states in the search space avoided and/or problems solving time.
5. *What is gained?* Past plans provide an outline of a solution to the new problem.

C. Reuse:

1. *What is reused?* Plans (past problem-solving cases).
2. *Why is it reused?* To guide the problem solver and avoid generative search.
3. *What properties does the reuse process place on the adaptation process?* The adaptation process must be capable of being guided by clipped partial plans
4. *How are the benefits measured?* In the number of states in the search space avoided and/or problems solving time
5. *What is gained?* Past plans provide an outline of a solution to the new problem.

D. Task:

1. What is the task? The domain?

The task is problem solving. The algorithm is domain-independent.

2. What are the inputs?

Problems specified in terms of initial and final states.

3. What are the outputs?

Solutions to the problems in the form of partial-order plan representations

4. What are the constraints on the outputs?

A correct solution must be found if one exists within the system's search-depth limits.

Are there characteristics of the domain that the method requires, relies on, or exploits in some way?

The method does not rely on any domain characteristics.

C. Evaluation:

1. What hypotheses were explored?

For a certain class of domains and problems, multi-plan adaptation will be more effective at reducing search than single-plan adaptation.

2. What type of evaluation?

Test cases were used to evaluate the algorithms to verify that they provided improvements. For these problems, we empirically tested problem solving methods (generative, single-plan, and multi-plan) against problem-solving time. This work is preliminary; a more complete evaluation is in progress.

3. What comparisons were made with other methods?

MPA was tested against a generative planner (SNLP) and a case-based reasoner (SPA) and showed improvements over both on the same problems.

4. How does the evaluation validate or illuminate your theory of adaptation of knowledge for reuse?

Our research provides an initial indication that multi-plan adaptation can be more effective than planning from scratch.

5. What are the primary contributions of your research?

Our research contributes an algorithm for multi-plan adaptation for case-based least-commitment planning and offers initial indications that multi-plan adaptation may be an efficient adaptation technique

References

Barret, A. & Weld, D. (1994) Partial order planning: Evaluating possible efficiency gains. Artificial Intelligence, 67(1), 71-112.

Cox, M. (1993) *Introspective Multistrategy Learning*. Cognitive Science Technical Report #2, Atlanta: Georgia Institute of Technology, College of Computing.

Fikes, Hart and Nilsson. (1972) STRIPS. *Artificial Intelligence*, pp 189-208, 1972.

Francis, A. and Ram, A. (1995). A Comparative Utility Analysis of Case-Based Reasoning and Control-Rule Learning Systems. In *Proceedings, ECML-95*, Heraklion, Crete, 1995. Available at ftp://ftp.cc.gatech.edu/pub/ai/students/centaur/ecml-95.ps.Z

Hanks, S., & Weld, D. (1992). Systematic Adaptation for Case-Based Planning. In *Proceedings of the First International Conference on AI Planning Systems*, 96-105. San Mateo, Ca: Morgan Kaufmann.

Hanks, S., & Weld, D. (1995). A Domain-Independent Algorithm for Plan Adaptation. Journal of Artificial Intelligence Research 2, p319-360

Hinrichs, T.R. (1992) *Problem Solving in Open Worlds: A Case Study in Design*. Lawrence Erlbaum, 1992.

Ihrig, L. & Kambhampati, S. (1994) Derivation Replay for Partial-Order Planning. In *Proceedings of AAAI-94*.

Kolodner, J.L. (1993). *Case-based Reasoning*. Morgan Kaufmann, 1993.

McAllester, D., & Rosenblitt, D. (1991). Systematic Nonlinear Planning. In Proceedings of the Ninth National Conference on Artificial Intelligence, 634-639. Menlo Park, Ca: AAAI.

Minton, S., Bresina, J., Drummond, M. (1994). Total order and Partial-Order Planning: A Comparative Analysis. *Journal of Artificial Intelligence Research 2*, p319-360

Redmond, M. (1992). Learning by observing and understanding expert problem solving. Georgia Institute of Technology, College of Computing Technical Report no. GIT-CC-92/43. Atlanta, Georgia.

Redmond, M. (1990). Distributed cases for case-based reasoning: Facilitating use of multiple cases. In Proceedings of AAAI-90. Cambridge, MA: AAAI Press/MIT Press.

Smyth, B. & Keane, M. (1995). Remembering to Forget: A Competence-Preserving Deletion Policy for CBR Systems. In *Proceedings of IJCAI-95*.

Weld, D. (1994). An introduction to least-commitment planning. *AI Magazine, (15) 4*, pages 27-61. Winter 1994.

Veloso, M. (1995). *Planning and Learning by Analogical Reasoning*. Springer-Verlag, 1995.

Veloso, M. & Blythe, J. (1994) Linkability: Examining causal link commitments in partial-order planning. In *Proceedings of the Second International Conference on AI Planning Systems*, p170-175, June 13-15 1994, Chicago, Illinois.

Learning to troubleshoot:

Multistrategy learning of diagnostic knowledge for a real-world problem-solving task*

Ashwin Ram

College of Computing

Georgia Institute of Technology†

S. Narayanan

Department of Biomedical and Human Factors Engineering

Wright State University‡

Michael T. Cox

College of Computing

Georgia Institute of Technology

*To appear in *Cognitive Science.*

†Address for correspondence: Dr. Ashwin Ram, College of Computing, Georgia Institute of Technology, Atlanta, Georgia 30332-0280. Phone: (404) 853-9372. Fax: (404) 853-9378. E-mail: `ashwin.ram@cc.gatech.edu`.

‡This research was performed while this author was at the School of Industrial and Systems Engineering, Georgia Institute of Technology.

Abstract

This article presents a computational model of the learning of diagnostic knowledge, based on observations of human operators engaged in a real-world troubleshooting task. We present a model of problem solving and learning in which the reasoner introspects about its own performance on the problem-solving task, identifies what it needs to learn to improve its performance, formulates learning goals to acquire the required knowledge, and pursues its learning goals using multiple learning strategies. The model is implemented in a computer system which provides a case study based on observations of troubleshooting operators and protocol analysis of the data gathered in the test area of an operational electronics manufacturing plant. The model not only addresses issues in human learning, but, in addition, is computationally justified as a uniform, extensible framework for multistrategy learning.

1 Introduction

The focus of our research is on the integration of different kinds of knowledge and reasoning processes into real-world systems that can learn through experience. In particular, we are interested in modeling active, goal-driven learning processes that underlie deliberative learning during the performance of complex reasoning tasks. This article presents a case study of multistrategy learning for the problem of learning diagnostic knowledge during a troubleshooting task. The case study is based upon observations of human operators engaged in this task. We present a computational model of problem solving and learning in which the reasoning system performs a diagnostic problem-solving task, and then introspects about its own performance on the task, identifies what it needs to learn to improve its performance, formulates learning goals to acquire the required knowledge, and pursues its learning goals using multiple learning strategies.

This research was motivated by two considerations. First, although there has been a significant growth of research on machine learning, much of this research has not been performed in the context of complex real-world problem-solving tasks (cf. Riddle, 1992). As a result, the issues of scalability and robustness of these methods, as they are applied to real-world problems, are still unresolved in many cases. To promote the applicability and usability of research methods, it is important to ground theories of reasoning, knowledge representation, and learning in the context of real-world tasks and domains.

Our second motivation was to provide a computational account of human learning in the context of a real-world problem. The model presented in this article is based on observations of troubleshooting operators and protocol analysis of the data gathered in the test area of an operational electronics manufacturing plant. The model is implemented in a computer system, Meta-TS,[1] which uses multiple types of knowledge to troubleshoot printed-circuit boards that fail in the test area of the manufacturing plant. Meta-TS has been evaluated on a series of troubleshooting problems, including actual problems encountered by the human operators in the manufacturing plant. The underlying model is intended as a computational model of human learning; in addition, it is computationally justified as a uniform, extensible framework for multistrategy learning in machine learning systems.

1.1 The problem

One of the critical areas in electronics assembly manufacturing is the test and repair area (Douglas, 1988; Kakani, 1987). It is estimated that about 20% of manufactured printed-circuit boards (PCBs) fail in the test area in an initial electronics assembly line, particularly in a medium-to-high variety product line when it takes time to achieve desired levels of process control. When PCBs spend a considerable amount of time in the test and repair area, it increases the work-in-process inventory and slows down the feedback to the manufacturing line necessary for achieving better process control. This results in significant deterioration of system performance. Computerized decision aids can potentially alleviate some of the major problems in the test and repair area and facilitate enhanced system performance. A key to developing computer-based aids is understanding the human problem-solving processes that carry out the complex task of troubleshooting in an assembly line situation. While there has been much interest in developing artificial intelligence (AI) applications in various areas of electronics manufacturing (e.g., Miller & Walker, 1988), most of this research has not dealt with the issues of learning or cognitive modeling.

It is generally accepted that learning is central to intelligent reasoning systems that perform realistic reasoning tasks, such as understanding natural language stories or solving complex problems (e.g., Anderson, 1987; Feigenbaum, 1963; Schank, 1983). It is impossible to anticipate all possible situations in advance and to hand-program a machine with exactly the right knowledge to deal with all the situations that it might be faced with. Rather, during the performance of any non-trivial reasoning task, whether by human or by machine, there will always be failures. An important aspect of intelligence lies in the ability to recover from such failures and, more importantly, to learn from them so as not to make the same mistake in future situations.

In the Meta-TS system, reasoning failures consist of incorrect troubleshooting diagnoses, no diagnosis (impasses), and successful diagnoses from inefficient problem-solving.[2] When such failures occur, the system must be able to select and apply an appropriate learning strategy in order to improve the chances of making a correct diagnosis in similar future situations. Thus, one approach a reasoning system might take is to reflect over the reasoning that went into making the original diagnosis and then use this introspective analysis to form a basis for selecting a learning strategy. To model this process theoretically, we have developed a computational model of introspective reasoning for decision-making about learning needs and associated learning strategies. This model is instantiated in the context of the diagnostic problem-solving task in the domain of electronics assembly manufacturing.

1.2 Multistrategy learning

Learning manifests itself in humans with multiple strategies over a multitude of learning problems. Over the past few years, research in machine learning and cognitive science has focused on the development of independent learning algorithms for many classes of these problems. Some of the algorithms that are tailored to particular learning problems include inductive learning (e.g., induction of decision trees (Quinlan, 1986), conceptual clustering (Fisher, 1987; Michalski & Stepp, 1983)), analytical learning (e.g., explanation-based learning (DeJong & Mooney, 1986; Mitchell, Keller, & Kedar-Cabelli, 1986), learning from explanation failures (Hall, 1988; VanLehn, Jones, & Chi, 1992)), and analogical learning (e.g., analogy (Falkenhainer, 1989; Gentner, 1989), case-based learning (e.g., Carbonell, 1986; Hammond, 1989)). Recently, under the banner of "multistrategy learning," there has been much interest in combining or otherwise integrating these and other learning methods in order to address more complex situations than does independent "monostrategy learning" (see, e.g., Michalski & Tecuci, 1994). Multistrategy learning systems use a variety of control methods to integrate and combine several learning strategies into a single computer model, providing power and flexibility over a wide range of problems.

An alternative approach to flexible learning is exemplified by cognitive architectures such as Soar (Laird, Rosenbloom, & Newell, 1986). Soar takes a broad approach to learning, using a single learning mechanism (chunking), rather than multiple learning strategies to account for learning on different classes of problems. Instead of explicit representations of different problem solving and learning methods and explicit selection between them, Soar is based on "weak methods" (universal subgoaling and chunking) from which higher-level strategies emerge. The system has been shown to model explanation-based generalization, strategy learning, macro-operator learning, learning from advice, and other kinds of learning (Steier et al., 1987).

Regardless of whether it is possible that a single underlying mechanism might be able to account for all these methods, however, it is still important to identify and study the methods themselves (and the conditions under which they are useful), particularly when developing computational models of human learning in which behaviors corresponding to these learning methods are exhibited. Rather than assume a uniform mechanism from which the strategies emerge, the multistrategy approach integrates separate learning strategies into a unified whole by providing a system with some mechanism for combining the strategies (or for choosing from among them). The desired learning behavior(s) can then be modeled by manipulating the suite of strategies available to the learner, by adjusting the manner of combination or the decision mechanism that chooses between strategies, or by changing the kinds of learning goals available for pursuit. One advantage of this approach is that different learning behaviors can be modeled directly and explicitly.

Our methodological stance is to develop an explicit theory of the different types of reasoning and learning that the system is to perform. We wish to understand the nature of various learning methods, the kinds of situations to which the methods apply, the kinds of knowledge that can be learned with them, and the limitations each method implies. Our approach uses a set of available learning strategies that are selected through an introspective analysis of the system's reasoning processes. Our method, called *introspective multistrategy learning*, combines metacognitive reasoning with multistrategy learning to allow the system to determine what it needs to learn and how that learning should be performed.

1.3 Introspective multistrategy learning

In order to fully integrate multiple learning algorithms into a single multistrategy system, it is beneficial to develop methods by which the system can make its own decisions concerning which learning strategies to use in a given circumstance. Often, knowledge about applicability conditions and utility of learning strategies is implicit in the procedures that implement the strategy; this further complicates the problem the system faces when automatically choosing a learning algorithm. Our solution to this problem is to represent knowledge of learning strategies and applicability conditions for these strategies explicitly in the system itself. An additional methodological benefit of this approach is that it requires the researcher to formulate such information as an explicit part of the proposed theory of learning, thus improving the specification of the theory.

In addition to the world model that describes its domain, an introspective multistrategy learning system has access to meta-models describing its reasoning and learning processes, the knowledge that this reasoning is based on, the indices used to organize and retrieve this knowledge, and the conditions under which different reasoning and learning strategies are useful. A meta-model is also used to represent the system's reasoning during a performance task, the decisions it took while performing the reasoning, and the results of the reasoning. All of this knowledge can then be used to guide multistrategy learning using introspective analysis to support the strategy selection process.

The introspective process in our model relies on meta-explanations about reasoning. These are similar to self-explanations (Chi & VanLehn, 1991; Pirolli & Bielaczyc, 1989; Pirolli & Recker, in press; VanLehn, Jones, & Chi, 1992), with the difference that self-explanations are explanations about events and objects in the external world, whereas our meta-explanations are explanations about events and objects in the reasoning system's train of thoughts—the mental world. While experimental results in the metacognition literature suggest that introspective reasoning can facilitate reasoning and learning (see, e.g., Schneider, 1985; Weinert, 1987; and the further review of the metacognition literature in section 5.1), it is important to develop computational models that specify the mechanisms by which this facilitation occurs and the kinds of knowledge that these mechanisms rely on.

Our approach is motivated by computational and system design considerations as well. The approach relies on a declarative representation of meta-models for reasoning and learning. There are several advantages of maintaining such structures in memory. Because these structures represent reasoning processes explicitly, the system can directly inspect the reasons underlying a given processing decision it has taken and evaluate the progress towards a goal. Thus, these representations can also be used to assign blame, to analyze why reasoning errors occurred, and to facilitate learning from these errors. Furthermore, these knowledge structures provide a principled basis for integrating multiple reasoning and learning strategies, and the unified framework makes it possible to incorporate additional types of learning situations and additional learning strategies for these situations.

The key representational entity in our learning theory is a *meta-explanation pattern* (Meta-XP), which is a causal, introspective explanation structure that explains how and why an agent reasons, and which helps the system in the learning task (Cox & Ram, 1992; Ram & Cox, 1994). There are two broad classes of Meta-XPs. *Trace Meta-XPs* record a declarative trace of the reasoning performed by a system, along with causal links that explain the decisions taken. The trace holds explicit information concerning the manner in which knowledge gaps are identified, the reasons why particular hypotheses are generated, the strategies chosen for verifying candidate hypotheses, and the basis for choosing particular reasoning methods for each of these. Trace Meta-XPs are similar to "reasoning traces" (Carbonell, 1986; Minton, 1988; Veloso & Carbonell, 1994) or "justification structures" (Birnbaum et al., 1990; deKleer et al., 1977; Doyle, 1979), with the difference that Trace Meta-XPs represent, in addition to the subgoal structure of the problem and justifications for operator selection decisions, information about the structure of the (possibly multistrategy) reasoning process that generated a solution. For example, at the highest level of granularity, a node in a Trace Meta-XP might represent the choice of a reasoning method such as association-based search or heuristic reasoning, and at a more detailed level a node might represent the process of selecting and using a particular association or heuristic. These structures could, therefore, be viewed as representing the "mental operators" underlying the reasoning process.

The major contribution of our approach, however, is the use of a new kind of meta-explanation structure to represent classes of learning situations along with the types of learning needed in those situations. This structure, called an *Introspective Meta-XP*, aids in the analysis of the reasoning trace to analyze the system's reasoning process, and is an essential component of a multistrategy learning system that can automatically identify and correct its own shortcomings. Thus, instead of simply representing a trace of the reasoning process, we also represent the knowledge required to analyze these traces in order to determine what to learn and how to learn it. As outlined in table 1, the system uses Introspective Meta-XPs to examine the declarative reasoning chain (recorded in step 0) in order to both explain the reasoning process and to learn from it after a problem-solving episode. These structures associate a failure type[3] (detected in step 1) with learning goals and the appropriate set of learning strategies for pursuing those goals. Thus, given a specific learning goal, as opposed to the failure itself, the system can explicitly plan for achieving that goal in it background knowledge (or even defer the goal pursuit until a later learning opportunity arises), much like traditional planners pursue goals in the world.[4]

Table 1 should be placed near here.

Therefore, an Introspective Meta-XP performs three functions: in step 2B it aids in blame assignment (determining which knowledge structures are missing, incorrect or inappropriately applied); in step 2C it aids in the formulation of appropriate learning goals to pursue; and in step 2D it aids in the selection of appropriate learning algorithms to recover and learn from the reasoning error. Such meta-explanations augment a system's ability to introspectively reason about its own knowledge, about gaps within this knowledge, and about the reasoning processes which attempt to fill these gaps. In Meta-TS, the use of explicit Meta-XP structures allows direct inspection of the need to learn that arises from a problem-solving failure, and of the bases for the selection of an appropriate learning strategy to address that need.

The remainder of this article is organized as follows. The next two sections present the technical details of the computational model, including the problem-solving system (section 2) and the introspective multistrategy learning system (section 3) that constitute the major components of the model. The article then discusses both a quantitative and a qualitative evaluation of the model (section 4) and relates the model to research in both artificial intelligence and psychology (section 5). The article concludes with pragmatic implications of the model in education (section 6) and a summary (section 7).

2 Diagnostic problem-solving

Before presenting the problem-solving component of the Meta-TS system, this section will describe the diagnostic problem-solving task addressed by Meta-TS, and the environment from which the human data was collected that constituted Meta-TS's problem set.

2.1 A real-world problem-solving task

NCR's manufacturing plant located near Atlanta has state-of-the-art facilities in electronics assembly manufacturing with a newly installed surface mount technology (SMT) line. Our project began when the plant became operational in January 1990 (Cohen, 1990). At that time, the plant was facing typical start-up problems experienced by most new facilities. There were a high number of printed-circuit boards (PCBs) in the test and repair region waiting to undergo troubleshooting.

resulting in high work-in-process inventories. Our analysis of the system revealed that developing a model of the troubleshooting operator would provide a structure for designing and implementing computer-based tools for this task. The effort would also facilitate formalization of the task, which could then be used in the design of instructional systems (Clancey, 1986), thereby facilitating the development of a flexible work force that is complementary to the policy of the company.

A schematic of the manufacturing plant is shown in figure 1. An unpopulated board enters the SMT line where it is populated with components, soldered, cleaned, and sheared. The populated board then enters the test and repair area. A typical PCB manufacturing line has two major test and repair areas, the "in-circuit test" area and the "functional test" area. PCBs are shipped only after they pass both the in-circuit test (ICT) area and the functional test area. Our research focuses on the troubleshooting process when a PCB fails in the ICT area.

Figure 1 should be placed near here.

In the ICT area, the populated printed-circuit board is mounted on an automated ICT machine. The ICT machine checks individual components as well as connections between components for proper functioning through several test procedures. If the PCB passes the tests, an appropriate message appears on the console of the ICT machine. If the PCB fails any of the tests, the ICT machine produces a ticket listing the detected failure(s). For example, a component on the PCB could fail to meet the desired specifications, known as the "nominal" values of the component's parameters. The ICT machine may also provide additional symptomatic information which can be used by the human operator in the troubleshooting process. The operator then uses the information in the ticket to troubleshoot the PCB. Troubleshooting is a complex task which can be broken into two components: diagnosis and repair. Diagnosis is a problem-solving task in which the operator arrives at a description of cause of the failure and identifies an appropriate set of repair actions to fix the faulty PCB. Repair involves carrying out the repair actions (in this case, usually a set of manual actions performed by the operator on the board).

We developed a computational model of an operator involved in the task of troubleshooting a faulty PCB. The model was based on protocol analysis of over 300 problem-solving episodes gathered in the ICT area of the NCR plant (Cohen, Mitchell, & Govindaraj, 1992), and implemented in a computer system that performed the troubleshooting task. Figure 2 shows part of an example troubleshooting protocol from Cohen (1990, p. 206). Of the data collected, one set (30%) was used in the development of the model and the remaining set (70%) was used to perform both behavioral validation and process validation of the computational model. We found that although the problem-solving model was a fair representation of a skilled troubleshooting operator, it had some limitations. First, the system assumed that its knowledge was correct and complete during the reasoning process. It is difficult to hand-code all the knowledge required for this task. Furthermore, even if this could be done, the system would still be faced with the "brittleness" problem. Due to the dynamics of the system state changes in the electronics manufacturing domain, the computational model must be flexible and robust. For example, one of the pieces of knowledge in the system was "Resistor r254 is often damaged." This occured due to a process problem in the manufacturing plant. If the process problem were fixed, the association would no longer be valid. The system must have the capability of altering its world model to reflect changes in the real world. In addition, the problem-solving model did not capture improvement in the problem-solving skills of the troubleshooting operator. Thus, the model was incomplete as a cognitive model of human troubleshooting.

Figure 2 should be placed near here.

These considerations motivated our research towards incorporation of a learning model in the system. The complete problem-solving and learning system is fully implemented in the Meta-TS program, which has been evaluated using the data gathered at the NCR plant. In this article, we will focus primarily on the learning aspects of the system; however, to provide context, we first describe the problem-solving module of Meta-TS.

2.2 The diagnostic problem-solving module

A schematic of the problem-solving system, the troubleshooting module of Meta-TS, is shown in figure 3 (Narayanan et al., 1992). The module takes as input the ICT ticket information and the PCB information. The output of the module is a diagnosis and a set of recommended repair actions. The problem-solving process uses various types of knowledge, troubleshooting actions, and control methods, briefly discussed below.

Figure 3 should be placed near here.

The problem-solving module uses various types of knowledge as well as available real-world troubleshooting actions to hypothesize the cause of a failure and to suggest repair actions for that failure. Based on the data from the human operators in the NCR plant, we categorized diagnostic knowledge into two broad types, associations and heuristics. *Associations* are simple rules which directly map a particular symptom to a specific diagnosis. The operator may perform an intermediate action to confirm the hypothesis, but usually does not perform a series of search sequences. This type of knowledge is context-sensitive and is indexed by board type. *Heuristics* are standard rules of thumb. These rules are not context-sensitive and are applicable across board types. Heuristics are used by the operator for troubleshooting when there is no known association for a given problem situation. This knowledge determines the series of standard operating procedures performed in troubleshooting a faulty PCB. Some examples of associative and heuristic knowledge in the system are shown in table 2.

Table 2 should be placed near here.

In addition to associative and heuristic knowledge, the problem-solving module can also use *troubleshooting actions*, which are intermediate subtasks performed by the system to gather the board information. These correspond to explicit operator actions used in gathering information and confirming intermediate hypotheses. Finally, the *control methods* in the problem-solving module are procedures that enable the system to look at the symptoms, utilize the appropriate type of knowledge, invoke proper intermediate actions, and finally arrive at the diagnosis result. This result, also called a "diagnostic," is a description of the failure along with the *repair action(s)* necessary to fix the faulty PCB. It is also possible that the ICT reading is incorrect (known in the industry as a "bogus" reading), in which case the PCB is rerun through the ICT machine. Some examples of troubleshooting actions, control methods, and repair actions are shown in table 3.

Table 3 should be placed near here.

The problem-solving module was initially implemented as a separate system, and then later incorporated into Meta-TS along with the introspective multistrategy learning module. The implementation used AT&T 2.1 C++ on a SUN workstation under the UNIX operating system. The ICT ticket information and the PCB information were represented as C++ classes. Class representations were also used for associative and heuristic knowledge, control methods, troubleshooting actions, and repair actions in the system. An example of a problem-solving episode is shown in figure 4.

Figure 4 should be placed near here.

In order to validate the troubleshooting process of the problem-solving module, we added an explanation facility to keep track of the system's problem-solving process and produce a trace of the problem solving at the end of each problem-solving episode. The problem-solving traces were compared with the verbal protocol data gathered from the human operators at the assembly plant. (These problem-solving traces also played a central role in the learning module; this will be discussed in more detail in section 3.) Using the problem-solving traces, the problem-solving module was validated on 75% of the problem-solving episodes in our data for a major category of board failures. On 84% of these episodes, the model arrived at the same diagnostic result as an operator did in the real world for the same input information; and on 68% of the episodes, similar actions were performed in the solution process (Narayanan et al., 1992). This remainder of this article focuses on the learning module; further details of the problem-solving module can be found in Cohen (1990) and Narayanan et al. (1992).

3 Learning diagnostic knowledge

Although the results from the stand-alone model of troubleshooting showed that the problem-solving module constituted a reasonably good model of a skilled troubleshooting operator, the model also contained obvious shortcomings. Electronics manufacturing, like most other real-world domains, is a complex and highly dynamic process. A complete model of troubleshooting in such a domain requires a large amount of knowledge; in addition, the operator's knowledge will be inherently incomplete and subject to change as the process being modeled changes. The problem-solving module, in contrast, was based on the assumption that the available knowledge was complete and correct; it did not have the flexibility necessary to deal with this task domain over an extended period of time. Furthermore, the model failed to capture the improvement of problem-solving skills through experience, an important aspect of human performance in any task domain.

For these reasons, we developed a learning module that allowed the system to learn incrementally from each problem-solving episode. This module was based on observations of troubleshooting operators in the plant, protocol analysis of the problem-solving process, and critical examination of the computational model of the troubleshooting operator as implemented in the problem-solving module. The overall system, called Meta-TS, uses multiple learning strategies, both supervised and unsupervised, and a strategy selection mechanism to invoke appropriate strategies in different situations. Supervised learning occurs in situations in which a novice troubleshooter receives explicit input from a skilled troubleshooter (the supervisor). In unsupervised learning, a troubleshooter adapts his or her domain knowledge based on problem-solving experience without expert input.

Since a particular problem-solving episode may involve several pieces of knowledge (potentially of different types), the troubleshooter, whether human or machine, must be able to examine the reasons for successes and failures during problem solving in order to determine what needs to be learned. For example, if the system fails to arrive at an correct diagnosis, it needs to determine which piece of knowledge was missing or incorrect. To effectively accomplish this, the system must be able to examine its own problem-solving processes. Thus, the problem-solving traces produced by the explanation facility discussed earlier are a crucial component of the computational model of learning.

Meta-TS uses declarative representations of the knowledge and methods used for problem-solving in order to facilitate critical self-examination. A trace of the problem-solving process is constructed during the troubleshooting episode, and introspectively analyzed during the learning phase to determine what the system might learn from that episode. The analysis also helps the system select the learning strategy appropriate for that type of learning.

3.1 What is to be learned?

Since the problem-solving module relies on associative and heuristic knowledge, the learning module must, in general, be able to acquire, modify, or delete such associations and heuristics through experience. In order to be more specific about the constraints on and output of the learning task, it is necessary to examine the troubleshooting model in more detail. Recent research in diagnostic problem solving has proposed the use of "deep" reasoning methods (Davis, 1985) or integration of "deep" and "shallow" reasoning methods in knowledge-based systems (Fink & Lusth, 1987) and in tutoring systems (Lesgold et al., 1988). Our observations revealed that operators rely predominantly on "shallow" reasoning methods using heuristic and context-sensitive associative knowledge during problem solving (Cohen, 1990; Cohen, Mitchell, & Govindaraj, 1992; Narayanan et al., 1992). This may be due to the fact that the ICT machine filters out most of the topographic knowledge of the PCB and causal knowledge of the components in the board through a series of tests. Maxion (1985) makes a similar observation about human problem-solving in the domain of hardware systems diagnosis, noting that "diagnostic judgement is based on gross chunks of conceptual knowledge as opposed to detailed knowledge of the domain architecture" [pp. 268-269]. The observation by Barr and Feigenbaum (1981), that humans often solve a problem by finding a way to think about the problem that facilitates the search for a solution, was clearly evident in our study. In this task domain, the search is carried out through "shallow" reasoning using associations and heuristics; furthermore, the search is sensitive to process changes and can sometimes make use of a human expert. Thus, the learning strategies implemented in Meta-TS focus on the supervised and unsupervised acquisition, modification, and deletion of associative knowledge through the analysis of reasoning traces that, however, do not contain detailed domain knowledge.

Associative knowledge improves the system performance in two ways. First, it improves the speed of the problem-solving process. Using associative knowledge typically results in the reduction of some intermediate steps in the reasoning process, thus resulting in some savings in the time required to troubleshoot; this is particularly significant if the problem-solving steps involve real-world actions (such as the lifted leg procedure) which take time to execute. This reduction is important for assembly line tasks which are typically highly time-constrained. Second, associative knowledge can provide solutions in cases where heuristic knowledge requires information about the board that is not easy to obtain. In general, associative knowledge contributes to the quality and correctness of the solution for a large number of the problem-solving situations. This was evident in our data from the electronics assembly plant, and has also been observed by other researchers (e.g., Arabian, 1989). Thus, an important type of learning is one in which the operator learns associations through experience.

Human operators involved in troubleshooting also appear to learn some heuristic knowledge. We noticed that the training program for novice human operators primarily focuses on manual skills such as soldering and performing actions such as "ohming out." However, the problem-solving process of skilled human operators in the plant revealed that they often use certain standard operating procedures or heuristics. The source of this heuristic knowledge appears to be the result of generalization of associations learned over time while troubleshooting. In addition, as is the case for associations, heuristics can be learned through both supervised and unsupervised learning methods. The current implementation of Meta-TS focuses on the learning of associative knowledge through experience and does not include strategies for learning heuristics. More research is needed to develop such strategies.

3.2 The introspective multistrategy-learning module

Our approach to multistrategy learning is based on the analysis of declarative traces of reasoning processes to determine what and how to learn (Ram & Cox, 1994). A particular troubleshooting episode may involve many different associations, heuristics, and troubleshooting actions. If the final diagnosis is incorrect, the system analyzes its reasoning process, assigns blame for its failure, and determines what it needs to learn in order to avoid repeating a similar mistake in the future. If the diagnosis is correct, the system can determine what it might learn in order to improve the process that led up to this diagnosis. Finally, depending on the type of learning that is necessary, the system must invoke an appropriate learning strategy. Thus, learning is viewed as a deliberative, planful process in which the system makes explicit decisions about what to learn and how to learn it (Hunter, 1990b; Quilici, in press; Ram, 1991; Ram & Hunter, 1992; Ram & Leake, in press; Redmond, 1992). In our introspective multistrategy learning framework, these decisions are based through introspective analysis of the system's performance, which relies on metaknowledge about the reasoning performed by the system during the performance task, about the system's knowledge, and about the organization of this knowledge (Ram & Cox, 1994; Ram, Cox, & Narayanan, in press).

The submodules and the control flow in the introspective multistrategy learning module are shown in figure 5 along with the sources of information used by the submodules. The problem-solving module has a declarative representation of the associative knowledge used in troubleshooting. The learning module can add, delete, or modify associative knowledge in the problem-solving module. It also has a set of verification actions and a set of declaratively represented learning strategies.

During a troubleshooting episode, a trace of the reasoning performed by the system along with causal links that explain the intermediate decisions taken is recorded in an instance of a Trace Meta-XP by the system's explanation facility. The explainer uses input from the problem-solving module in the form of actions taken, knowledge used to make decisions, and the diagnosis outcome. It also uses the ICT ticket reading and its representation of the PCB from the world model. From this input it reconstructs the reasoning trace and passes it to the introspector.

Figure 5 should be placed near here.

After every problem-solving episode, the introspector examines the reasoning trace and uses information gathered from tests on the world to determine if the system can learn something from this experience. Learning occurs when the system fails to make the correct diagnosis (due to missing or incorrect knowledge) or when the system ascertains that the problem-solving process can be made more efficient. The tests also help to generate and verify hypotheses that explain why the reasoning which produced the diagnosis failed, and play a role similar to the real-world actions performed by experimentation systems (e.g., Carbonell & Gil, 1990; Rajamoney, 1989). Specifically, in addition to accessing ICT information and PCB information which is provided as input to the system, the system uses troubleshooting actions to gather additional information about the PCB and verification actions to obtain statistical information and to gather information from an expert troubleshooter.

Finally, based on what needs to be learned, an appropriate learning strategy is triggered, which results in the modification of existing knowledge in the problem-solving system. The learning module contains a set of learning strategies represented along with information for strategy selection. In Meta-TS, the introspector is implemented as a C++ class with methods for each learning strategy (see figure 6); this class encodes the knowledge that corresponds to the Introspective Meta-XPs discussed earlier. The learning strategies currently implemented in Meta-TS are discussed in the next section.

Figure 6 should be placed near here.

3.3 Learning strategies

Meta-TS has several strategies for learning associative knowledge for the troubleshooting task, including unsupervised knowledge compilation, supervised learning from an expert, postponement of learning goals, and forgetting invalid associations. Each strategy requires us to make several design decisions; these are discussed below. All the strategies discussed below are fully implemented.

3.3.1 Unsupervised learning

The first strategy is that of unsupervised, incremental inductive learning, which creates an association when the problem-solving module arrives at a correct diagnosis using heuristic knowledge. The introspector compiles the heuristic knowledge into an association using a learning method similar to knowledge compilation (Anderson, 1989). The motivation for this type of learning is performance gain through reduction of the number of intermediate steps when the system encounters a similar problem in the future, although use of this strategy also reinforces correct problem-solving sequences.

An example of the unsupervised learning of associations through experience is shown in figure 7. In this example, the ICT ticket reading indicated that the resistor component r22 had failed with a measured reading of 16 ohms. The nominal reading of this component, from the PCB specification, is 20 ohms. The problem-solving module reads the symptom (step 1 in the figure). It first tries to find an association that directly maps the observed symptoms into a diagnosis, but fails to find one (step 2). It then finds (step 3) and invokes (step 4) a heuristic that recommends performing the troubleshooting action "ohming out" on r22. The action is performed in step 5, but it finds that r22 is not faulty. Finally, the system outputs the diagnosis that the ICT ticket reading was "bogus."

These steps are stored in a Trace Meta-XP, which is analyzed after the troubleshooting is complete. In this example, the introspector performs additional tests on the PCB to determine that the diagnosis is correct. Since this is an experience in which a correct diagnosis was reached through the use of heuristic knowledge in a situation for which no association existed, an Introspective Meta-XP recommends that a new association be learned: "If the ICT ticket indicates that r22 has failed, and the measured reading is slightly lower than the nominal value, then output the diagnosis that the ICT ticket is "bogus." This association is installed in the system and is used for future problem solving; it may also be deleted later if it is incorrect or becomes obsolete (e.g., if the problem is fixed).

Figure 7 should be placed near here.

Several design decisions were made in our implementation of this learning strategy:

- *What is the right time to activate the strategy?* Unsupervised learning takes place at the end of a troubleshooting episode. This strategy is activated when Meta-TS arrives at the right solution using heuristic knowledge alone.

- *When is it useful to form an association?* Meta-TS uses statistical information about the episode (e.g., the number of steps involved in problem solving) and determines if there will be performance gain through the reduction in the number of intermediate steps while troubleshooting a similar board. This information is only used to determine whether learning a new association would speed up the troubleshooting process, and does not ensure that the learned association is "correct."

- *What is the right association to learn?* Consider the situation when the ICT input is I, the intermediate steps are 1, 2, 3, 4, and 5, and the diagnostic result is O. Meta-TS would form an association either between I and O, or between I, the final step (step 5 in this example), and O. Domain knowledge is used to decide between the two alternatives. Our data shows that human operators typically form an association between the input and output without any intermediate steps when the diagnostic result is "Bogus ICT ticket reading." In contrast, when the operator decides to replace a defective part, he or she is conservative and performs either a visual inspection or some other intermediate action to confirm the hypothesis. Meta-TS behaves in a similar manner.

Discussion: We observed that human operators used yellow tags ("PostIt notes") to note down a recurring problem, especially when they believe that this information will be useful in the future. This happened when they performed several intermediate steps during troubleshooting, and typically after they had arrived at the diagnostic result. This was the motivation for including this learning strategy, and also the basis for the first two design decisions.

3.3.2 Supervised learning

The second learning strategy creates a new association through supervisory input. This strategy is triggered when the system arrives at an incorrect solution using heuristic and/or associative knowledge. The system attempts to acquire a correct associative knowledge from a skilled troubleshooter (the "supervisor"). This mechanism is similar to the interactive transfer of expertise in TEIRESIAS (Davis, 1979). However, the knowledge learned in our system is not in the form of production rules, but in the form of frames and slots for association records.

An example of the supervised learning of associations through experience is shown in figure 8. In this example, the ICT ticket reading indicates that the resistor component r24 has failed, the measured reading of 21.2 ohms being much higher than the nominal value of 10 ohms. There are no known associations for this problem, so the system applies a heuristic that recommends "ohming out" on r24. In this example, "ohming out" confirms that the ticket reading was correct. Another heuristic recommends a simple visual inspection of the PCB, which shows that r24 is missing from this PCB. This is output as the diagnosis from the problem-solving module. The introspector in the learning module finds that the diagnosis is not correct; in this case, there is a missing IC component, u37, that is responsible for the problematic ICT ticket reading. The expert supervisor suggests that a new association be formed that, for this input, recommends performing a visual inspection on u37. This association is learned and installed for future use.

Figure 8 should be placed near here.

Several design decisions were made in our implementation of this learning strategy:

- *What is the right time to activate the learning strategy?* This strategy is activated at the end of the troubleshooting episode when the system arrives at an incorrect solution or is unable to make any inference based on the information available to it.
- *What is the structure of the supervisory input?* The structure of the desired supervisory input is determined by the manner in which the associative knowledge is stored in the system. In contrast, the conversation between an expert and novice troubleshooter is not so structured. Since the relevant information transmitted between them is domain- and task-oriented, however, that structure is exploited in the dialogs used by Meta-TS. While the current implementation of this learning strategy does not model the full richness of a troubleshooter's interactions with an expert, the more structured interaction allows an objective evaluation of the model. The user interaction in the current implementation of the system is very simple since that was not the focus of our research; however, it would be relatively easy to include a more sophisticated dialog system if desired.
- *How can the system reason about the validity of the expert input?* This is an open question for learning systems in general. However, for our purposes, the input from the expert troubleshooter can be assumed to be correct. Meta-TS does not critically examine whether the input given by the expert is correct; it directly takes the associative knowledge input by the expert and adds it to its knowledge base.

Discussion: Novice operators ask expert troubleshooters such as engineers or highly trained technicians when they have problems in their task. We use the expert-novice metaphor for the supervisor-system interaction. The system learns the knowledge input by the supervisor (as do novice troubleshooters). The interaction between Meta-TS and the expert is capable of gathering the relevant associative knowledge. However, the actual mode of communication does not reflect expert-novice interaction in the real world. For example, Meta-TS currently does not model apprenticeship relationships in troubleshooting (e.g., Redmond, 1992).

The improvement in system performance from this learning strategy depends on the quality and validity of the expert input. The new knowledge is subject to change, depending on the future episodes encountered by the system. If the new knowledge obtained from supervisory input is found to be reliable in a number of future instances, the confidence in the gained knowledge is increased. However, if the new knowledge is incorrect, it is deleted over time (see section 3.3.4). Thus, the transfer of knowledge is immediate but the "sustainability" of the knowledge depends on the use of the gained knowledge.

3.3.3 Postponement

A third learning strategy is that of postponement (Hammond et al., 1993; Ram, 1991). This strategy is triggered when the system is unable to get immediate input from a skilled troubleshooter. The system posts a learning goal (Ram, 1991; Ram & Hunter, 1992; Ram & Leake, in press), keeps track of the reasoning trace for the particular problem-solving episode, and asks questions at a later time to gather appropriate associative knowledge. Postponement takes place when there is no supervisory input at the end of a troubleshooting episode. The learning goal and the trace of the troubleshooting episode are stored in the introspector. Suspended learning goals can be satisfied both through supervised or unsupervised methods at a later time.

At the beginning of a new troubleshooting episode, the introspector checks whether the reasoning trace associated with any suspended learning goal is based on an input problem that is similar to the current problem. Similarity is determined based on the fault type indicated on the ICT ticket and the difference between the nominal and measured readings. If one or more matching learning goals are found and an expert is available, the introspector triggers a question-and-answer session by presenting the information it has on the past episodes. Details of the episodes are presented only if the supervisor desires to look at it. If expert input is obtained, new associative knowledge is added to the system and the resolved learning goals are deleted along with the associated reasoning traces. The system then continues to solve the current problem using the new associative knowledge.

If no expert input is available, the introspector tries to solve the current problem. If it succeeds, the learning goals that matched this problem are automatically satisfied without supervisory input. As before, these goals and associated reasoning traces are deleted since the system is now capable of solving those problems. The system is also capable of solving similar problems in the future with the newly formed associative knowledge.

Again, several design decisions were made in our implementation of this learning strategy:

- *What is the appropriate time for question-answer sessions?* A question-answer session takes place either at the end of a troubleshooting episode or at the beginning of a new episode. Question-answer sessions are not needed for learning goals that become redundant when new associative knowledge is learned without user input. There are, of course, several other factors involved in deciding when to ask a question, including sociological factors such as the personalities of and interpersonal interactions between the troubleshooter and the expert technician; these are outside the focus of our model.

- *How should the suspended question be presented?* Meta-TS uses context-sensitive presentation of information. When the user is asked for input in a situation which matches a similar situation that is associated with a learning goal suspended from a prior episode, the information in the reasoning traces leading to that learning goal is presented to provide a context for the dialog. Using the principle of progressive disclosure, the user can ask to examine more details.

- *When are learning goals active?* Learning goals are always "active" in the sense that any problem-solving episode or question-answer session could contain the information sought by a prior learning goal; however, learning goals are not actively pursued by the system until the desired information is available in the available input, at which time the algorithm that carries out the learning is executed.

Discussion: Novice operators seek input from the expert supervisor when they are unable to find the solution to a problem. Operators may ask for input when a similar new problem is encountered. Undiagnosed PCBs may also be stored and retrieved later for re-analysis, which corresponds to the deferment of a learning goal until a later opportunity to get the appropriate information is encountered. The design decision to present prior reasoning traces to the expert is intended to facilitate user interaction; although operators can often recall what they did in earlier situations, it is arguable whether they remember all the details of the entire troubleshooting process for the earlier situations.

3.3.4 Forgetting

Two additional learning strategies delete associative knowledge when it is no longer valid. These strategies are primarily targeted at the brittleness problem that is encountered when the manufacturing process is changed and existing associations are rendered obsolete. The first strategy uses expert input to delete associations, and is invoked at the end of every problem-solving episode. The system queries the supervisor to determine whether any associations used in the reasoning trace of that episode should be deleted. If the supervisor has knowledge about, for example, a process change and the system dynamics has resulted in an association becoming obsolete, that information can be input to Meta-TS. This strategy works quite well in general, although it is, of course, dependent on the availability and quality of user input.

The second deletion strategy is unsupervised and does not require user input. This strategy is selected when Meta-TS arrives at an incorrect solution (as determined through additional tests on the PCB or through expert input) and the reasoning trace shows that a single association was used in arriving at the solution. Since heuristic knowledge in this task domain tends to be relatively stable, an incorrect diagnosis involving several heuristics and a single association is blamed on the association. The introspector tracks down this association and deletes it. The current implementation of this strategy cannot deal with situations in which more than one association is used; such situations require assigning blame to the particular association that was at fault.

Several design decisions were made in our implementation of this learning strategy:

- *Under what conditions should an association be deleted?* When the expert troubleshooter indicates that an association needs to be deleted, Meta-TS follows the supervisory input. In the unsupervised mechanism, the system behaves conservatively in the sense that a piece of associative knowledge is deleted only if the diagnostic result is incorrect and only one association was involved in the problem-solving process. In the current implementation, a user-definable parameter determines how many times an association needs to be responsible for an incorrect diagnosis before it is deleted; while not a general solution to the problem of determining when a piece of knowledge is no longer valid, this method is reasonable in our task domain given the highly dynamic nature of the manufacturing process. Empirical studies showed good performance with this parameter set to 1; hence, in the evaluations presented in section 4, the system was configured to delete an association if it led to a single incorrect diagnosis, but a different setting could be chosen if desired. Another learning strategy (not currently implemented) would be to make the association more specific so as to exclude the current situation.

- *What is the right time to activate the strategies?* Deletion of existing associative knowledge in Meta-TS takes place at the end of a troubleshooting episode. At this point, the system has available to it the trace of its reasoning process and also information about the correctness of its diagnostic result. Both are required in order to identify and delete incorrect knowledge.

Discussion: When the manufacturing process changes, it impacts the quality of the boards produced, the types of malfunctions that can occur, and consequently the operator troubleshooting. For example, let us assume that r243 is a known defective part, say, due to a poor quality vendor. When the vendor is changed, the part r243 may no longer be defective. Typically, this information is communicated from the manufacturing process line or when the operator recognizes the change in the situation due to a failure of the troubleshooting process. The first situation corresponds to the supervisory input case, and the second to the unsupervised case. It is arguable whether human operators can "forget" an association instantaneously; however, trained operators often stop using an obsolete association even if they do not actually "forget" it. The cognitive plausibility of various forgetting mechanisms is still an open research issue, although the methods implemented in Meta-TS are effective in dealing with the particular task at hand.

4 Evaluation

Meta-TS has been evaluated both qualitatively and quantitatively. We were interested both in comparing the results to the human data, as well as evaluating it as a machine learning system.

We evaluated the system using 42 actual problem-solving episodes gathered at the plant over a 2-month period (Cohen, 1990). The problems dealt with various types of resistor failures and are representative of the types of problems encountered over the 2-month period. To evaluate the learning methods, we tested the following five conditions on the 42 test problems.

- H (hand-coded): The original non-learning system with hand-coded associations. This condition represents a troubleshooting system that has been hand-designed by an expert researcher, and is useful as a benchmark in determining the strengths and limitations of the learning strategies.
- NL (no learning): The system with all associations removed and learning turned off. This condition represents a base case against which to evaluate the efficacy of the learning strategies; it uses only heuristic knowledge.
- L (learning): The system with all associations removed and learning turned on. This is the basic Meta-TS system with no prior experience.
- L42: The system with all associations removed, then trained it on the 42 test problems with learning turned on. The system was then evaluated by re-running it on the same 42 problems. This condition was intended to validate the learning strategies in Meta-TS by ensuring that they learned the knowledge required to solve the problems.
- L20: The system with all associations removed, then trained on 20 randomly generated training problems with learning turned on. The problems can be classified as easy, medium, and hard, based on degree of difficulty as measured using the number of intermediate steps in the troubleshooting process. We generated 20 random problems with the probabilities that the problem generated was easy, medium or hard set to 0.6, 0.2 and 0.2, respectively. The randomly generated training set is representative of the problems a human operator encounters over about a month at the job, both in terms of number and degree of difficulty. The problems varied from 42 test problems in various ways. In order to test the statistical significance of the results, several independent random training problems were generated. "L20" in the following discussion and in figures 8 through 12 indicates the mean L20 value at various data points.

Each of these conditions were evaluated quantitatively for speed and accuracy on the 42 test problems, and also qualitatively by examining the content of the learned knowledge and details of the solution process. For supervised learning strategies, we provided "expert" input to the system based on what was appropriate to the input problem and domain experience.

4.1 Quantitative evaluation

Two quantitative performance measures were used: the accuracy of the diagnostic result, and the speed (measured by the number of intermediate problem-solving steps) of arriving at the diagnosis. Figures 9 through 13 illustrate the system performance over the 42 problems for the H, NL, L, L42 and L20 conditions.

Diagnostic accuracy: Figure 9 shows the cumulative accuracy of the system for the various conditions. The H condition arrived at the correct diagnosis in 86% of the 42 problems. The L42 condition arrived at the correct diagnosis in 81% of the problems. The values for the L20. L. and NL conditions were 76.8%, 76%, and 71% respectively. The graphs illustrate both these final accuracy figures, as well as the improvement of the system with experience.

Figure 9 should be placed near here.

Figures 10 and 11 compare the accuracy of the learning conditions relative to that of the hand-coded condition and non-learning conditions, relatively. By measuring the ratio, we compensate for differences in the intrinsic difficulty of the individual problems. Again, the graphs illustrate both the final result as well as improvement with experience. The ratio of the L42 condition to that of the H condition is about 0.94; for L20 and L conditions, the ratios are 0.9 and 0.89, respectively. As compared with the NL condition, the L42 condition is about 1.14 times more accurate; for L20 and L, the ratios are 1.08 and 1.07, respectively.

A t-test was performed to test the null hypothesis that the NL performance was equal to the mean L20 performance. During this analysis, 5 independent random L20 sets were used; their mean was tested against a constant, which is the value of NL. Using the operating characteristics curve, we determined that for the variance observed in the data, the sample size of 5 was sufficient to keep the type II error (β) within 0.10. The t-test showed that the difference between NL and L20 is statistically significant. At the end of 42 episodes, $t(4) = 3.04$, $p < 0.05$ for the null hypothesis L20 = NL. With only 35 episodes, the statistical advantage of L20 over NL was only marginally significant after sequential Bonferoni adjustment ($t(4) = 3.33, p = 0.03$). Thus, the learning system showed improvement in performance as compared to the non-learning system, and this improvement was statistically significant after 42 training episodes.

An independent t-test was performed to compare the mean L20 performance to the performance in the H condition. The test showed that the performance of the learning system was poorer than the performance of the system using hand-coded associations after 15 episodes at a type I error (α) value of 0.05. Thus, the learning in Meta-TS was better than the NL condition, but poorer than the H condition.

Figure 10 should be placed near here.

Figure 11 should be placed near here.

Speed of problem solving: Figures 12 and 13 compare the speed of the solution process (measured by the number of intermediate steps) with the various learning conditions relative to the hand-coded and non-learning condition, respectively. The L20 and L42 conditions consistently arrive at the diagnostic result faster than the H condition. The L condition takes about 20 problem episodes to reach the same speed as that of the H condition and then consistently arrives at the diagnostic result faster than the H condition. At the end of the 42 problem episodes, the ratios of the learning conditions to the hand-coded conditions are: 1.52 (L42 to H), 1.24 (L20 to H), and 1.06 (L to H). In comparison to the non-learning version of the program, all the three learning conditions, L42, L20, and L, consistently arrived at the diagnostic result faster than the NL condition. At the end of the 42 problem episodes, the ratios of the learning conditions to the hand-coded conditions are: 1.75 (L42 to H), 1.41 (L20 to H), and 1.20 (L to H).

Figure 12 should be placed near here.

Figure 13 should be placed near here.

Discussion: The results of the quantitative evaluation can be summarized as follows. The multistrategy learning module in Meta-TS clearly contributes to enhanced system performance in the troubleshooting task; this improvement is statistically significant. In comparison with the non-learning system with no hand-coded associations, the associative knowledge learned by Meta-TS increases the accuracy of the diagnostic result and speeds up the problem-solving process. The performance of Meta-TS further increases when it is trained on similar problems before it is applied to novel problems. The associative knowledge learned by Meta-TS enables it to arrive at the same solution as that of the system with the hand-coded associative knowledge between 89% and 94% of the time.

Although Meta-TS is faster than the hand-coded version, it was also seen that Meta-TS with the various learning strategies did not outperform the system with the hand-coded associations in terms of the accuracy of diagnostic result. We hypothesize that it may be due to two reasons. First, in order not to spoon-feed the system and possibly invalidate the results, the supervisory input given to the system throughout the evaluation process was kept very minimal. Thus, the expert input to the system for either the 20 or 42 problem-solving episodes may not have enabled Meta-TS to obtain all the associations that an operator in the plant obtains over a period of several months of task performance.[5] Second, the currently implemented system does not contain all the learning strategies that a human operator uses. However, given the learning architecture used in Meta-TS, it is possible to incorporate additional learning strategies, once identified, in the system.

4.2 Qualitative evaluation

We also evaluated Meta-TS using various qualitative metrics. We compared the learned associations with the hand-coded associations, the solution process of a human operator to that of Meta-TS on the same problems, and the methods and knowledge used by Meta-TS to troubleshoot and learn to those used by human operators. The results are as follows.

Quality of the learned associative knowledge: We compared the associative knowledge learned by Meta-TS while troubleshooting the 42 test boards to the hand-coded associations in the original problem-solving system. Meta-TS learned 33% of the hand-coded associations. It was unable to learn some of the hand-coded associations as it did not encounter them in the training or test problem set. (Recall that the hand-coded associations were based on over 300 problem-solving episodes.) Meta-TS also learned other associations that did not correspond to the hand-coded ones which enabled it to perform better in terms of the speed of the solution process.

Comparison of the solution process: We compared the process of arriving at a solution in Meta-TS and operator troubleshooting processes from the verbal protocols. The L20 condition was used in this comparison as it best represents a fairly trained operator because of the training input discussed earlier. We divided the problems into two sets. Difficult problems included those in which Meta-TS was unable to arrive at the correct solution or those which required several intermediate problem-solving steps; in about 50% of these problems, human operators also spent a considerable time in troubleshooting. The remaining problems were considered easy for Meta-TS; in about 80% of these, human operators also arrived at the correct solution fairly quickly.

Comparison of troubleshooting knowledge and learning processes: As discussed earlier, human operators rely predominantly on shallow reasoning methods using heuristic and context-sensitive associative knowledge in this task domain. This is modeled through the use of heuristic and associative knowledge in the troubleshooting model. Furthermore, humans operators learn associative knowledge through experience in the task by several means. They may obtain input from expert troubleshooters. They may also notice recurring instances of a problem and may then form an association between the input and the diagnostic result. In some situations, when immediate input from an expert is not available, they may place the PCB with the ICT reading aside and then attempt to obtain supervisory input at a later time when they encounter a similar problem. All these means of learning associative knowledge by human operators are reflected by the various learning strategies in Meta-TS, as discussed earlier. The strategies are integrated through the introspective learning architecture of the system.

4.3 Generality of the model

In addition to evaluating Meta-TS itself, we also need to evaluate the generality and flexibility of the underlying model of introspective multistrategy learning. We are performing another case study in a task domain that is very different from the one discussed in this article. The Meta-AQUA system, presented in Ram and Cox (1994), uses "deep" causal knowledge to understand natural language stories. The performance task in this system that of causal and motivational analysis of conceptual input in order to infer coherence-creating structures that tie the input together. Meta-AQUA is an introspective multistrategy learning system that improves its ability to understand stories consisting of sequences of descriptions of states and actions performed by characters in the real world. The system is based on the AQUA system (Ram, 1991, 1993), which is a computational model of an active reader. Meta-AQUA uses the same theory of introspective multistrategy learning to allow the system to recover from, and learn from, several types of reasoning failures through an introspective analysis of its performance on the story understanding task.

In both AQUA and Meta-AQUA, reading is viewed as an active, goal-driven process in which the reasoning system focuses attention on what it needs to know and attempts to learn by pursuing its goals to acquire information (Ram, 1991). Such a system models the hypothetical metacognitive reader discussed by Weinert (1987), who "perceives a gap in his knowledge, . . . attempt[s] to take notes on the relevant information, to understand it," undertakes "learning activities from a written text," examines "how his assessment of his own knowledge structures compares with his expectations about the demands" of an uncoming performance task, and can tell us about his "preferred learning strategies, and his evaluation of his own situation and the possible consequences" [p. 7]. While reasoning in this task domain is very different from the often shallow diagnostic processes used in assembly line manufacturing, and the two use very different kinds of knowledge, it is possible to use the same model of introspective multistrategy learning in both task domains (Ram, Cox, & Narayanan, in press). Although further details of Meta-AQUA are outside the scope of this article, we introduce the system here as further computational evidence of the generality of our approach.

4.4 Limitations of the model

While we have achieved a reasonable degree of success in modeling human troubleshooters as they learn and gain experience on an assembly line, our model also has several limitations. Some of these limitations are due to the level of granularity of the introspective multistrategy learning theory; this issue is discussed further towards the end of this section and in section 5.2. Here, we discuss limitations in our use of the theory as a computational model of human troubleshooting, including limitations arising from the computational framework used to develop Meta-TS, and limitations due to the current implementation of the Meta-TS program.

Implementational limitations are, perhaps, the least important. For example, our current implementation of the method for interactive transfer of expertise during supervised learning is very simple. We were interested in the integration of multiple learning methods into a single system and not so much in developing new learning algorithms; if better learning algorithms were developed, they could be incorporated into Meta-TS with relative ease. Similarly, the implementation of forgetting simply involves deletion of an association; clearly, human forgetting is a much more complex process (e.g., Cox, 1994). Other such simplifications have been pointed out in the preceding technical discussion. It is interesting to note, however, that Meta-TS can model many aspects of the human data even with these simplifications.

Meta-TS is also limited in certain ways as a computational model of human troubleshooting. Our model focuses on ICT troubleshooting operators who routinely work on testing and repair, and does not model technicians or engineers who are, for example, called in to help with this task on certain occasions, such as when a very difficult problem is encountered. Although expert technicians and engineers may also rely on associative and heuristic knowledge similar to that observed in our study, they may also use other kinds of knowledge, such as topographic models or causal knowledge. For example, Hale (1992) shows that humans use both weak causal heuristics and domain-specific knowledge in learning symptom-fault associations in causal domains. Senyk, Patil, and Sonnenberg (1989) argue that in medical diagnosis experienced diagnosticians apply a variety of reasoning techniques, ranging from the association of symptoms and diseases to causal principles about diseases and first-principle analysis grounded in basic science. Based on research in process control, maintenance, and medicine, Rasmussen (1993) outlines the importance of causal knowledge related to the mental model of human operators during problem solving. While the Meta-TS framework permits extension of the model, the current model does not represent these kinds of knowledge. Consequently, the model is limited to situations when the shallow reasoning methods are sufficient and may not be directly useful for situations where "deeper" knowledge of the domain is necessary (for example, situations when the root cause of the problem is to be found).

Another limitation of the current model is the simplified view of the troubleshooter's interaction with the environment. This interaction not only includes expert-novice interaction in supervised learning situations, but also includes interaction with the equipment and artifacts in the environment that the troubleshooter is situated in. In particular, our model focuses on cognitive processing and not on situated interactions; while the former is important, the relationship between the two is an important issue for future research.

Finally, while the learning strategies used in Meta-TS are similar to those used by a typical "trained" operator, and the overall learning behavior of Meta-TS is also comparable with that of a human operator, our analysis does not provide a detailed comparison with human thought processes on individual problems. In particular, on a given set of problems, we have neither shown that an individual human operator formulates the particular reasoning traces that Meta-TS does, nor that he or she selects the particular learning strategies that Meta-TS does on each problem in that set. Such a comparison is extremely difficult since the specifics of a reasoning trace, and the corresponding choice of a learning strategy, depend on the domain knowledge and level of expertise of the troubleshooter, the prior problems encountered, the availability of an human expert, and other details. Furthermore, it is unclear how one could obtain protocols of human troubleshooters that specified their reasoning traces or their strategy selection decisions in sufficient detail to permit direct comparison on individual problem-solving episodes at the level of granularity of the computational model.[6] Thus, Meta-TS should be viewed as a model of a typical troubleshooting operator in a typical assembly line environment, and not as a detailed model of a specific individual operator solving a specific set of problems.

5 Discussion and related research

Diagnostic problem-solving has been studied by several researchers in cognitive science, artificial intelligence, psychology, and human-machine systems engineering. Specifically, there has been much work on troubleshooting in real-world domains, including that of Bereiter and Miller (1989) in computer-controlled automotive manufacturing, Govindaraj and Su (1988) in marine power plants, Katz and Anderson (1987) in program debugging, Kuipers and Kassirer (1984) in medicine, Maxion (1985) in fault-tolerant hardware systems, and Rasmussen (1984) in industrial process control. Much of this work is based on studies of human problem-solving. Rouse and Hunt (1984) discuss various models of operator troubleshooting based on experimental studies in simulated fault diagnosis tasks and present implications for training and aiding operators in these tasks. Research in artificial intelligence has resulted in computational models of knowledge-based diagnosis (e.g., Chandrasekaran, 1988) and qualitative reasoning (e.g., deKleer & Williams, 1987).

A detailed review of research in human troubleshooting and diagnostic problem-solving is outside the focus of this article, which is concerned with issues in learning and introspection. In the remainder of this section, we will summarize related issues from the artificial intelligence and psychology literatures.

5.1 Artificial intelligence, metareasoning and multistrategy learning

There are several fundamental problems to be solved before we can build intelligent systems capable of general multistrategy learning, including: determining the cause of a reasoning failure (blame assignment), deciding what to learn (learning goal formulation), and selecting the best learning strategies to pursue these learning goals (strategy selection). We claim that a general multistrategy learning system that can determine its own learning goals and learn using multiple learning strategies requires the ability to reflect or introspect about its own reasoning processes and knowledge. Pollock (1989) distinguishes between knowledge about the facts that one knows and knowledge about one's motivations, beliefs and processes. Introspective multistrategy learning is based on the both kinds of metaknowledge; we argue that introspective access to explicit representations of knowledge and of reasoning processes is essential in making decisions about what and how to learn.

One form of introspection that has been implemented in many systems is the use of reasoning traces to represent problem-solving performance; an early example of this approach was Sussman's (1975) HACKER program. Reasoning trace information has primarily been used for blame assignment (e.g., Birnbaum et al., 1990) and for speedup learning (e.g., Mitchell, Keller, & Kedar-Cabelli, 1986). In addition, we propose that such information, suitably augmented with the kinds of knowledge represented in our Introspective Meta-XP structures, can be used as the basis for the selection of learning strategies in a multistrategy learning system.

Many research projects in AI have demonstrated the advantages of representing knowledge about the world in a declarative manner. Similarly, our research shows that declarative knowledge about reasoning can be beneficial. The approach is novel because it allows strategy selection systems to reason about themselves and make decisions that would normally be hard-coded into their programs by the designer, adding considerably to the power of such systems. Meta-reasoning has been shown to be useful in planning and understanding systems (e.g., Stefik, 1981; Wilensky, 1984). Our research shows that meta-reasoning is useful in multistrategy learning as well. To realize this ability, our model incorporates algorithms for learning and introspection, as well as representational methods using which a system can represent and reason about its meta-models.

From the artificial intelligence point of view, our approach is similar to other approaches based on "reasoning traces" (e.g., Carbonell, 1986; Minton, 1988) or "justification structures" (e.g., Birnbaum et al., 1990; deKleer et al., 1977; Doyle, 1979), and to other approaches that use characterizations of reasoning failures for blame assignment and/or multistrategy learning (e.g., Mooney & Ourston, 1991; Park & Wilkins, 1990; Stroulia & Goel, 1992). A major difference between these approaches and ours is our use of explicit representational structures (Introspective Meta-XPs) to represent classes of learning situations along with the types of learning needed in those situations, a type of knowledge that is crucial in multistrategy learning systems. Other types of knowledge may also be important in multistrategy learning systems. For example, Pazzani's (1991) OCCAM system has generalized knowledge about physical causality that is used to guide multistrategy learning. In contrast, we propose specific knowledge about classes of learning situations that can be used to guide learning strategy selection. Integration of these and other approaches is still an open research issue.

Approaches to multistrategy learning fall into four broad categories, which we call strategy selection models, toolbox models, cascade models, and single mechanism models. The common element in all these approaches is the use of multiple learning methods to allow the reasoning system to learn in multiple types of learning situations.

In *strategy selection models*, the reasoning system has access to several learning strategies, each represented as a separate algorithm or method. Learning involves an explicit decision stage in which the appropriate learning strategy is identified, followed by a strategy application stage in which the corresponding algorithm is executed. Methods for strategy selection also differ. Pazzani's (1991) OCCAM system, for example, tries each learning strategy in a pre-defined order until an applicable one is found; Reich's (1993) BRIDGER system uses a task analysis of the problem-solving task to determine the appropriate learning strategies for each stage of the task; Hunter's (1990a) INVESTIGATOR system represents prerequisites for application of each learning strategy; and Ram and Cox's (1994) Meta-AQUA system uses characterizations of reasoning failures to determine what to learn and, in turn, the learning strategies to use to learn it.

Toolbox models are similar to strategy selection models in that they too incorporate several learning strategies in a single system. The difference is that these strategies are viewed as tools that can be invoked by the user to perform different types of learning. The tools themselves are available for use by other tools; thus, learning strategies may be organized as coroutines. An example of this approach is Morik's (1991) MOBAL system, in which learning occurs through the cooperation of several learning tools with each other and with the user. Another example of the toolbox class is the PRODIGY system (Carbonell, Knoblock, & Minton, 1991). The system combines explanation-based learning, case-based (analogical) learning, abstraction, experimentation, static analysis, and tutoring. However, the system is designed as a research test-bed for analyzing and comparing various methods, rather than as a system that chooses a learning method itself. Instead, the experimenter chooses a learning module to run against a given problem-solving test suite.[7]

In *cascade models*, two or more learning strategies are cascaded sequentially, with the output of one strategy serving as the input to another. For example, Danyluk's (1991) GEMINI system uses a cascade of explanation-based learning, conceptual clustering, and rule induction strategies, in that order, to combine analytical and empirical learning into a single learning system. Clearly, these categories of models are not exclusive of each other (e.g., a strategy selection system may choose to cascade learning strategies in certain circumstances), but they serve to characterize the major ways in which learning strategies may be integrated.

Finally, *single mechanism models* use a single underlying mechanism as a "weak method" which can perform different types of learning depending on the situation. Examples of such models are Laird, Rosenbloom and Newell's (1986) SOAR, and Tecuci and Michalski's (1991) MTL. These approaches are sometimes contrasted with multistrategy approaches in that, although they provide multiple methods for learning when characterized at a theoretical level, only a single learning algorithm is implemented in the computer model. As discussed earlier, however, it is still important to characterize the learning strategies that are implemented by (or that emerge from) the single mechanism, and the circumstances under which different strategies are used by the system, even in such systems as those above.

Our approach is an example of a strategy selection model. To develop a computer program that can deal with the complexities of real-world troubleshooting, the system must deal with an incomplete world model, dynamic changes in the world which renders part of the world model obsolete, and multiple forms of knowledge (much of it shallow). This requires the integration of multiple learning methods (inductive, analytical, and interactive) in both supervised and unsupervised situations. Our experience with the Meta-TS system shows that a strategy selection architecture can deal effectively with such problems. Furthermore, our approach provides a general framework for integrating multiple learning methods. The learning strategies are not dependent on the domain, but are, however, dependent on the types of knowledge used in the performance task.

5.2 Psychology, metacognition and human learning

Much of the metaknowledge research in artificial intelligence has focused on knowledge about knowledge, or knowledge about the facts that one does or does not know (e.g., Barr, 1979; Davis, 1979; Davis & Buchanan, 1977). Much of the metacognition research in psychology has also focused on similar issues, in particular, on cognitive processes, strategies, and knowledge having the self as referent. Of particular interest is psychological research on metamemory which includes, in addition to knowledge about knowledge, knowledge about memory in general and about the peculiarities of one's own memory abilities (Weinert, 1987). The empirical results obtained from the Meta-TS system support the claim that metaknowledge should also include knowledge about reasoning and learning strategies.

Experimental results in the metacognition literature suggest that introspective reasoning can facilitate reasoning and learning. For example, Delclos and Harrington (1991) report that subject conditions with general problem-solving skill training and those with both problem-solving and metacognitive skill training demonstrate equal performance on a logical problem-solving task. With greater task complexity, however, subjects with the problem-solving and metacognitive training exhibit greater performance than either a control group or the group with problem-solving training alone. Swanson (1990) establishes the independence of general problem aptitude from metacognitive ability. Subjects with relatively high metacognitive ability, but low aptitude, often compensate for low aptitude by using metacognitive skills so that their performance is equivalent to subjects with higher aptitude. Our research extends these results by specifying computational mechanisms for metacognitive processing, focusing in particular on the selection and use of learning strategies.

There are at least three important ways that metacognitive knowledge and capabilities bear on work in introspective learning. First, and foremost, is the emphasis on cognitive self-monitoring. This behavior is a human's ability to read their own mental states during cognitive processing (Flavell & Wellman, 1977; Nelson & Narens, 1990; Wellman, 1983). Thus, there is a moment-by-moment understanding of the content of one's own mind, and an internal feedback for the cognition being performed and a judgement of progress (or lack thereof). Psychological studies have confirmed a positive effect between metamemory and memory performance in cognitive monitoring situations (Schneider, 1985; Wellman, 1983). This directly supports the hypothesis that there must be a review phase when reasoning or a parallel review process that introspects to some degree about the performance element in a cognitive system.

Second, our Meta-XP theory places a heavy emphasis on explicit representation. Trains of thought, as well as the products of thought, are represented as metaknowledge structures, and computation is not simply calculated results from implicit side-effects of processing. This emphasis echoes Chi's (1987) argument that to understand knowledge organization and to examine research issues there must be some representational framework. Although diverging from the framework suggested by Chi, Meta-XP theory provides a robust form to represent knowledge about knowledge and process. For example, Meta-XPs can represent the difference between remembering and forgetting (Cox, 1994; Cox & Ram, 1992). Since forgetting is the absence of a successful retrieval (i.e., a mental event which did not occur), forgetting is difficult to represent in most frameworks. An explicit representation of it, however, has been formulated in the Meta-AQUA system mentioned earlier, and used to reorganize memory indexes when forgetting occurs. Moreover, forgetting is an important issue in additional machine learning (Markovitch & Scott, 1988) and cognitive psychology (Mensink & Raaijmakers, 1988; Wellman & Johnson, 1979) research. Meta-TS implements a simple form of forgetting in which obsolete knowledge is deleted once it is identified.

Finally, because the approach taken by the introspective learning paradigm clearly addresses the issue of memory organization, it can assign blame to errors that occur from mis-indexed knowledge structures and poorly organized memory. Although Meta-TS does not need to deal directly with the mis-indexed knowledge problem,[8] extensions of this approach to other types of tasks and domains may need to do so, particularly if deep knowledge is required. Memory organization of suspended goals, background knowledge, and reasoning strategies is as important in determining the cause of a reasoning failure as are the goals, propositions and strategies themselves (Ram, Cox, & Narayanan, in press). Thus, memory retrieval and encoding issues are relevant in deciding what to learn and which learning strategy is appropriate. This claim is supported by the metamemory community's focus on organizational features of memory and their relation to the human ability to know what they know, even in the face of an unsuccessful memory retrieval. Extending the Meta-TS model to include a cognitive model of human memory (including memory organization) is an important issue for future research.

One of the major differences between the manner in which humans learn and that in which machines do is that humans perform dynamic metacognitive monitoring or self-evaluation. Humans often know when they are making progress in problem solving, even if they are far from a solution, and they know when they have sufficiently learned something with respect to some goal (Weinert, 1987). They know how to allocate mental resources and can judge when learning is over. Many of the above reviews (e.g., Chi, 1987; Schneider, 1985; Wellman, 1983) cite evidence for such claims. Research in Meta-XP theory is a step in the direction in adding this metacognitive monitoring capability to AI systems, but this is beyond the capabilities of the present implementation of Meta-TS.

It should be noted that the learning strategies represented in Meta-TS, or other strategy selection programs such as Meta-AQUA, are at a finer level of granularity than those examined by much of psychology. For example, it would be misleading to assert that the types of learning strategies studied by the metacognition community are similar to index learning, explanation-based generalization, and other learning strategies used in Meta-AQUA, although Meta-TS's strategies are closer in content to the cognitively plausible learning methods suggested by Anderson (1989) and others. Instead, metacognition research focuses on a person's choice of strategic behaviors at the level of cue elaboration, category grouping, and target rehearsal (in memory tasks); re-reading of text, question generation, and keyword search (in text interpretation tasks); or solution checking, saving intermediate results in an external representation, and comprehension monitoring (in problem-solving tasks). However, many of the results from research on metacognition do support the overall approach taken in this paper, that of using introspection to support the selection of appropriate strategies in different situations. Although we are currently building computer systems at what might be called the micro-level, it would be eventually be desirable to build systems that integrate the kinds of behavior exhibited by human learners at the macro-level as well.

Finally, we would like to emphasize that our model of learning is agnostic about the issue of "consciousness." Weinert (1987) argues convincingly that consciousness is a persistent unsolved problem in metacognition. However, we make no claims about when people are aware of their introspection, nor that active, strategic learning necessarily implies a conscious process. We would expect some of the processing in our model to be deliberative and conscious, especially when the reasoning system becomes aware of a failure in its reasoning process, but it is evident that people possess and use metacognitive knowledge that they are sometimes not aware of. This issue is beyond the scope of and orthogonal to the point of this article; the computational model presented here may be used to take an intentional stance (Dennett, 1987) towards the learning process in which the competence of the learner is modeled using goals, learning decisions, learning actions, and so forth as the basic theoretical constructs, independent of the degree of conscious self-awareness of these processes in human thought.

6 Pragmatic implications of the model for education

While Meta-TS is intended as a model of learning, our results have several pragmatic implications for the design of interactive learning environments. Major issues in developing an intelligent tutoring system include what to teach and how to teach; specific points of importance are the student model, the teacher model, the organization of knowledge, the simulation of the task, and the interface to the learner (Psotka, Massey, & Mutter, 1988; Spohrer & Kleiman, 1992). Our research suggests that it would be valuable to teach shallow troubleshooting knowledge, including context-specific associative knowledge and general heuristic knowledge. Furthermore, since our model of learning involves reasoning about actual troubleshooting experiences, and active pursuit of identified learning goals through multiple learning strategies, we suggest that novice troubleshooters be placed in simulated or actual problem-solving situations and encouraged to reason about what they are doing and why they are doing it. This approach is consistent with recent approaches suggested in the educational literature. For example, in Scardamalia and Bereiter's (1991) Teacher C model, the teacher is concerned with helping students formulate their own goals, do their own activation of prior knowledge, ask their own questions, direct their own inquiry, and do their own monitoring of comprehension. Redmond (1992) suggests a similar approach to learning through apprenticeship. His model is implemented in the CELIA system, which observes an expert troubleshooter (in this case, a car mechanic) solving the given problem, reasons explicitly about how it would solve the same problem, and determines what it needs to learn in order to be able to explain and predict the expert's behavior based on the differences between the expert's problem-solving processes and its own.

Several researchers have proposed simulation environments in which students play roles that are connected to their goals, and whose successful completion requires acquisition of the skills to be taught (e.g., Schank et al., 1994; Shute, Glaser, & Raghavan, 1988; van Berkum et al., 1991). Van Berkum and his colleagues, for example, identify four aspects of the design of such systems: simulation models, learning goals, learning processes, and learning activity. In their model, students pursue learning goals with three dimensions: the type of knowledge, the representation of that knowledge, and the generality and applicability of that knowledge. Learning occurs through interaction with simulated environments using four types of learning actions (orientation, hypothesis generation, testing, and evaluation) which are guided by the learning goals. The learning model implemented in Meta-TS provides a basis for the design of such learning environments. In particular, we suggest that these environments provide facilities to encourage students to introspect, question, and explore. Exploring the relationship between learning and education is a fruitful direction for future research.

7 Conclusions

We have presented a computational framework for introspective multistrategy learning, which is a deliberative or strategic learning process in which a reasoner introspects about its own performance to decide what to learn and how to learn it. The reasoner introspects about its own performance on a reasoning task, assigns credit or blame for its performance, identifies what it needs to learn to improve its performance, formulates learning goals to acquire the required knowledge, and pursues its learning goals using multiple learning strategies. In this article, we have presented a model of human troubleshooting based on this framework, focusing in particular on the learning aspects of the model. The model is implemented in a computer program which models human troubleshooters and also provides a case study in the use of the computational framework for the design of multistrategy machine learning systems. Our approach relies on a declarative representation of meta-models for reasoning and learning. The resulting computational model represents a novel combination of metacognition and multistrategy learning and provides a framework for cognitive modeling as well as the design of artificial intelligence systems.

In this article, we have presented a particular case study of an introspective multistrategy learning system for the complex task of diagnostic problem-solving on the assembly line of a real-world manufacturing plant. The research was based on observations of troubleshooting operators and protocol analysis of the data gathered in the test area of an operational electronics manufacturing plant. The model was implemented in a computer system, Meta-TS, which uses multiple types of knowledge to troubleshoot printed-circuit boards that fail in the test area. Meta-TS was evaluated on a series of troubleshooting problems, including actual problems encountered by the human operators in the manufacturing plant. The results were evaluated both qualitatively and quantitatively to determine the efficacy of the learning methods as well as to compare the model to human data. The results show that the model can be computationally justified as a uniform, extensible framework for multistrategy learning, and cognitively justified as a plausible model of human learning.

Acknowledgements

Research for this article was supported by the National Science Foundation under grant IRI-9009710, by the Air Force Office of Sponsored Research under contract F49620-94-1-0092, and by the Georgia Institute of Technology. We wish to thank Rick Alterman, Mike Byrne, Alan Lesgold, Mike Pazzani, John Pollock, Mimi Recker, Kurt VanLehn, and anonymous reviewers for their extremely helpful comments on various drafts of this article. The problem-solving model presented in this paper benefited from discussions with Sally Cohen, Christine Mitchell, T. Govindaraj, Ashok Goel, and Leon McGinnis at Georgia Tech, and was based on data collected by Sally Cohen. We thank Jim Wallace, Don Haywood, Mike Bunger, and Kenneth Mitchell of NCR's manufacturing plant in Atlanta for their cooperation during this study.

8 References

Anderson, J. (1987). Skill acquisition: Compilation of weak-method problem solutions. *Psychological Review, 94*:192–210.

Anderson, J. (1989). A theory of the origins of human knowledge. *Artificial Intelligence, 30*:313–351.

Arabian, J. (1989). *Computer integrated electronics manufacturing and testing*. New York: Mercel Dekker.

Barr, A. (1979). Meta-knowledge and cognition. In *Proceedings of the Sixth International Joint Conference on Artificial Intelligence* (pp. 31–33).

Barr, A. & Feigenbaum, E. A. (Eds.). (1981). *The handbook of artificial intelligence* (Vol. 1). Reading, MA: Addison-Wesley Publishing Company.

Bereiter, S. R. & Miller, S. M. (1989). A field-based study of troubleshooter's information utilization in computer-controlled manufacturing systems. *IEEE Transactions on Systems, Man, and Cybernetics, 19*(1):205–219.

Birnbaum, L., Collins, G., Freed, M., & Krulwich, B. (1990). Model-based diagnosis of planning failures. In *Proceedings of the Eighth National Conference on Artificial Intelligence* (pp. 318–323). Cambridge, MA: The MIT Press.

Carbonell, J. G. (1986). Derivational analogy: A theory of reconstructive problem solving and expertise acquisition. In Michalski, R. S., Carbonell, J. G., & Mitchell, T. M. (Eds.), *Machine learning II: An artificial intelligence approach* (pp. 371–392). San Mateo, CA: Morgan Kaufman Publishers, Inc.

Carbonell, J. G. & Gil, Y. (1990). Learning by experimentation: The operator refinement method. In Kodratoff, Y. & Michalski, R. S. (Eds.), *Machine learning III: An artificial intelligence approach* (pp. 191–213). San Mateo, CA: Morgan Kaufman Publishers, Inc.

Carbonell, J. G., Knoblock, C. A., & Minton, S. (1991). PRODIGY: An integrated architecture for planning and learning. In VanLehn, K. (Ed.), *Architectures for intelligence: The twenty-second Carnegie Mellon symposium on cognition* (pp. 241–278). Hillsdale, NJ: Lawrence Erlbaum Associates, Inc.

Chandrasekaran, B. (1988). Generic tasks as building blocks for knowledge-based systems: The diagnosis and routine design examples. *Knowledge Engineering Review, 3*:183–219.

Chi, M. T. H. (1987). Representing knowledge and metaknowledge: Implications for interpreting metamemory research. In Weinert, F. E. & Kluwe, R. H. (Eds.), *Metacognition, motivation, and understanding*. Hillsdale, NJ: Lawrence Erlbaum Associates, Inc.

Chi, M. T. H. & VanLehn, K. A. (1991). The content of physics self-explanations. *The Journal of the Learning Sciences, 1*(1):69–105.

Clancey, W. J. (1986). *Knowledge-based tutoring: The GUIDON program*. Cambridge, MA: The MIT Press.

Cohen, S. M. (1990). *A model of troubleshooting in electronics assembly manufacturing* (Tech. Rep. No. CHMSR 90-3). Unpublished master's dissertation, Georgia Institute of Technology, Center for Human-Machine Systems Research, Atlanta.

Cohen, S. M., Mitchell, C. M., & Govindaraj, T. (1992). Analysis and aiding the human operator in electronics assembly. In Helander, M. & Nagamachi, M. (Eds.), *Design for manufacturability: A systems approach to concurrent engineering and ergonomics* (pp. 361–376). London: Taylor and Francis.

Cox, M. T. (1994). Machines that forget: Learning from retrieval failure of mis-indexed explanations. In *Proceedings of the Sixteenth Annual Conference of the Cognitive Science Society* (pp. 225–230). Hillsdale, NJ: Lawrence Erlbaum Associates, Inc.

Cox, M. T. & Ram, A. (1992a). An explicit representation of forgetting. In *Proceedings of the Sixth International Conference on Systems Research, Informatics and Cybernetics*, Windsor, Ontario, Canada: The International Institute for Advanced Studies in Systems Research and Cybernetics.

Cox, M. T. & Ram, A. (1992b). Multistrategy learning with introspective meta-explanations. In *Machine Learning: Proceedings of the Ninth International Conference*. San Mateo, CA: Morgan Kaufman Publishers, Inc.

Cox, M. T. & Ram, A. (1994a). Failure-driven learning as input bias. In *Proceedings of the Sixteenth Annual Conference of the Cognitive Science Society* (pp. 230–236). Hillsdale, NJ: Lawrence Erlbaum Associates, Inc.

Cox, M. T. & Ram, A. (1994b). Choosing learning strategies to achieve learning goals. In *Proceedings of the AAAI Spring Symposium on Goal-Driven Learning* (pp. 12–21). Menlo Park, CA: AAAI Press.

Danyluk, A. P. (1991). Gemini: An integration of analytical and empirical learning. In Michalski, R. S. & Tecuci, G. (Eds.), *Proceedings of the First International Workshop on Multistrategy Learning* (pp. 191–206). Fairfax, VA: George Mason University, Center for Artificial Intelligence.

Davis, R. (1979). Interactive transfer of expertise: Acquisition of new inference rules. *Artificial Intelligence, 12*:121–157.

Davis, R. (1985). Diagnosis via causal reasoning: Paths of interaction and the locality principle. In Richardson, J. J. (Ed.), *Artificial Intelligence in Maintenance* (pp. 102–122). Park Ridge, New Jersey: Noyes Publications.

Davis, R. & Buchanan, B. G. (1977). Meta-level knowledge: Overview and applications. In *Proceedings of the Fifth International Joint Conference on Artificial Intelligence*, Cambridge, MA.

DeJong, G. F. & Mooney, R. J. (1986). Explanation-based learning: An alternative view. *Machine Learning, 1*(2):145–176.

deKleer, J., Doyle, J., Steele, G. L., & Sussman, G. J. (1977). Explicit control of reasoning. *SIGPLAN Notices, 12*(8).

deKleer, J. & Williams, B. (1987). Diagnosing multiple faults. *Artificial Intelligence, 32*:97–130.

Delclos, V. R. & Harrington, C. (1990). Effects of strategy monitoring and proactive instruction on children's problem-solving performance. *Journal of Educational Psychology, 83*:35–42.

Dennett, D. (1987). *The Intentional Stance*. Boston, MA: Bradford Books/MIT Press.

Douglas, P. N. (1988). Effective functional test design increases PCB throughput. *Electronic Manufacturing, 34*:16–18.

Doyle, J. (1979). A truth maintenance system. *Artificial Intelligence, 12*:231–272.

Falkenhainer, B. (1989). *Learning from Physical analogies: A study in analogy and the explanation process*. PhD thesis, University of Illinois, Department of Computer Science, Urbana, IL.

Feigenbaum, E. A. (1963). The simulation of verbal learning behavior. In Feigenbaum, E. A. & Feldman, J. (Eds.), *Computers and thought* (pp. 297–309). New York: McGraw-Hill Book

Company, Inc. Originally published during 1961 in Proceedings of the Western Joint Computer Conference, 19:121-132.

Fink, P. K. & Lusth, J. C. (1987). Expert systems and diagnostic expertise in the mechanical and electrical domains. *IEEE Transactions on Systems, Man and Cybernetics*, SMC-17(3):340–347.

Fisher, D. (1987). Knowledge acquisition via incremental conceptual clustering. *Machine Learning*, 2:139–172.

Flavell, J. H. & Wellman, H. M. (1977). Metamemory. In R. V. Kail, J. & Hagen, J. W. (Eds.), *Perspectives on the development of memory and cognition*. Hillsdale, NJ: Lawrence Erlbaum Associates, Inc.

Gentner, D. (1989). Mechanisms of analogical learning. In Vosniadou, S. & Ortony, A. (Eds.), *Similarity and analogical reasoning*. London: Cambridge University Press.

Govindaraj, T. & Su, Y. D. (1988). A model of fault diagnosis performance of expert marine engineers. *International Journal of Man-Machine Studies, 29*:1–20.

Hale, C. (1992). *Effects of background knowledge on associative learning in causal domains*. Unpublished doctoral dissertation, Georgia Institute of Technology, School of Psychology, Atlanta.

Hall, R. J. (1988). Learning by failing to explain: Using partial explanations to learn in incomplete or intractable domains. *Machine Learning, 3*:45–78.

Hammond, K., Converse, T., Marks, M., & Seifert, C. M. (1993). Opportunism and learning. *Machine Learning, 10*(3):279–309.

Hammond, K. J. (1989). *Case-based planning: Viewing planning as a memory task*. Perspectives in artificial intelligence. Boston: Academic Press.

Hunter, L. E. (1990a). Knowledge acquisition planning for inference from large datasets. In Shriver, B. D. (Ed.), *Proceedings of the Twenty Third Annual Hawaii International Conference on System Sciences* (pp. 35–45). Los Alamitos, CA: IEEE Computer Society Press.

Hunter, L. E. (1990b). Planning to learn. In *Proceedings of the Twelfth Annual Conference of the Cognitive Science Society* (pp. 261–268). Hillsdale, NJ: Lawrence Erlbaum Associates, Inc.

Kakani, T. (1987). Testing of surface mount technology boards. In *Proceedings of the International Test Conference* (pp. 614–620). Los Alamitos, CA: IEEE Computer Society Press.

Katz, I. & Anderson, J. R. (1987). Debugging: An analysis of bug-location strategies. *Human-Computer Interaction, 3*:351–399.

Kuipers, B. and Kassirer, J. (1984). Causal reasoning in medicine: Analysis of a protocol. *Cognitive Science, 8*:363–385.

Laird, J. E., Rosenbloom, P. S., & Newell, A. (1986). Chunking in Soar: The anatomy of a general learning mechanism. *Machine Learning, 1*:11–46.

Lesgold, A., Lajoie, S., Bunzo, M., & Eggan, G. (1988). Sherlock: A coached practice environment for an electronics troubleshooting job. In Larkin, J., Chabay, R., & Scheftic, C. (Eds.), *Computer assisted instruction and intelligent tutoring systems: Establishing communication and collaboration*. Hillsdale, NJ: Lawrence Erlbaum Associates, Inc.

Markovitch, S. & Scott, P. D. (1988). The role of forgetting in learning. In *Proceedings of the Fifth International Conference on Machine Learning* (pp. 459–465). San Mateo, CA: Morgan Kaufman Publishers, Inc.

Maxion, R. A. (1985). Artificial Intelligence approaches to monitoring system integrity. In Richardson, J. J. (Ed.), *Artificial Intelligence in Maintenance* (pp. 260–276). Park Ridge, New Jersey: Noyes Publications.

Mensink, G. & Raaijmakers, J. (1988). A model for interference and forgetting. *Psychological Review, 95*:434–455.

Michalski, R. S. & Stepp, R. E. (1983). Learning from observation: Conceptual clustering. In Michalski, R. S., Carbonell, J. G., & Mitchell, T. M. (Eds.), *Machine learning I: An artificial intelligence approach* (pp. 331-364). Los Altos, CA: Morgan Kaufman, Inc.

Michalski, R. S. & Tecuci, G. (Eds.). (1994). *Machine learning IV: A multistrategy approach*. San Francisco: Morgan Kaufman Publishers, Inc.

Miller, R. K. & Walker, T. C. (1988). *Artificial intelligence applications in the computer/electronics industry*. Madison, GA: SEAI Technical Publications/Fairmont Press.

Minton, S. (1988). *Learning effective search control knowledge: An explanation-based approach* (Tech. Rep. No. CMU-CS-88-133). Unpublished doctoral dissertation, Carnegie-Mellon University, Computer Science Department, Pittsburgh. .

Mitchell, T. M., Keller, R., & Kedar-Cabelli, S. (1986). Explanation-based generalization: A unifying view. *Machine Learning, 1*(1):47–80.

Mooney, R. J. & Ourston, D. (1991). A multistrategy approach to theory refinement. In Michalski, R. S. & Tecuci, G. (Eds.), *Proceedings of the First International Workshop on Multistrategy Learning* (pp. 115–130). Fairfax, VA: George Mason University, Center for Artificial Intelligence.

Morik, K. (1991). Balanced cooperative modeling. In Michalski, R. S. & Tecuci, G. (Eds.), *Proceedings of the First International Workshop on Multistrategy Learning* (pp. 65–80). Fairfax, VA: George Mason University, Center for Artificial Intelligence.

Narayanan, S., Ram, A., Cohen, S. M., Mitchell, C. M., & Govindaraj, T. (1992). Knowledge-based diagnostic problem solving and learning in the test area of electronics assembly manufacturing. In *Proceedings of the SPIE Conference on Applications of AI X: Knowledge-Based Systems*. Orlando, FL.

Nelson, T. O. & Narens, L. (1990). Metamemory: A theoretical framework and new findings. *The Psychology of Learning and Motivation, 26*:125–141.

Park, Y. & Wilkins, D. C. (1990). Establishing the coherence of an explanation to improve refinement of an incomplete knowledge base. In *Proceedings of the Eighth National Conference on Artificial Intelligence* (pp. 511–516). Cambridge, MA: The MIT Press

Pazzani, M. J. (1991). Learning to predict and explain: An integration of similarity-based, theory-driven and explanation-based learning. *The Journal of the Learning Sciences, 1*(2):153–199.

Pirolli, P. & Bielaczyc, K. (1989). Empirical analyses of self-explanation and transfer in learning to program. In *Proceedings of the Eleventh Annual Conference of the Cognitive Science Society* (pp. 450–457). Hillsdale, NJ: Lawrence Erlbaum Associates, Inc.

Pirolli, P. & Recker, M. (in press). Learning strategies and transfer in the domain of programming. *Cognition and Instruction, 12*.

Pollock, J. L. (1989). A general theory of rationality. *Journal of Theoretical and Experimental Artificial Intelligence, 1*:209–226.

Psotka, J., Massey, L. D., & Mutter, S. A. (Eds.). (1988). *Intelligent tutoring systems: Lessons learned*. Hillsdale, NJ: Lawrence Erlbaum Associates, Inc.

Quilici, A. (in press). Toward automatic acquisition of an advisory system's knowledge base. *Applied Intelligence*.

Quinlan, J. R. (1986). Induction of decision trees. *Machine Learning, 1*:81–106.

Rajamoney, S. (1989). *Explanation-based theory revision: An approach to the problems of incomplete and incorrect theories*. PhD thesis, University of Illinois, Department of Computer Science, Urbana, IL.

Ram, A. (1991). A theory of questions and question asking. *The Journal of the Learning Sciences, 1*(3&4):273–318.

Ram, A. (1993). Indexing, elaboration and refinement: Incremental learning of explanatory cases. *Machine Learning, 10*:201–248.

Ram, A. & Cox, M. T. (1994). Introspective reasoning using meta-explanations for multistrategy learning. In Michalski, R. S. & Tecuci, G. (Eds.), *Machine learning IV: A multistrategy approach* (pp. 349–377). San Francisco: Morgan Kaufman Publishers, Inc.

Ram, A., Cox, M. T., & Narayanan, S. (in press). Goal-driven learning in multistrategy reasoning and learning systems. In Ram, A. & Leake, D. B. (Eds.), *Goal-driven learning*, chapter 17. Cambridge, MA: The MIT Press/Bradford Books.

Ram, A. & Hunter, L. (1992). The use of explicit goals for knowledge to guide inference and learning. *Applied Intelligence, 2*:47–73.

Ram, A. & Leake, D. B. (in press). Learning, goals, and learning goals. In Ram, A. & Leake, D. B. (Eds.), *Goal-driven learning*. Cambridge, MA: The MIT Press.

Rasmussen, J. (1984). Strategies for state identification and diagnosis in supervisory control tasks, and design of computer-based support systems. *Advances in Man-Machine Systems Research, 1*:139–193.

Rasmussen, J. (1993). Diagnostic reasoning in action. *IEEE Transactions on Systems, Man, and Cybernetics, 23*:981–992.

Redmond, M. (1992). *Learning by observing and understanding expert problem solving* (Tech. Rep. No. GIT-CC-92/43). Unpublished doctoral dissertation, Georgia Institute of Technology, College of Computing, Atlanta.

Reich, Y. (1994). Macro and micro perspectives of multistrategy learning. In Michalski, R. S. & Tecuci, G. (Eds.), *Machine learning IV: A multistrategy approach.* San Francisco: Morgan Kaufman Publishers, Inc.

Riddle, P., (Ed.). (1992). *Proceedings of the Workshop on Integrated Learning in Real-World Domains, Ninth International Machine Learning Conference*, Aberdeen, Scotland.

Rouse, W. & Hunt, R. (1984). Human problem solving in fault diagnosis tasks. *Advances in Man-Machines Systems Research, 1*:195–222.

Scardamalia, M. & Bereiter, C. (1991). Higher levels of agency for children in knowledge building: A challenge for the design of new knowledge media. *The Journal of the Learning Sciences, 1*(1):37–68.

Schank, R. C. (1983). The current state of AI: One man's opinion. *The AI Magazine, 4*(1):3–8.

Schank, R. C., Fano, A., Jona, K., & Bell, B. (1994). The design of goal-based scenarios. *The Journal of the Learning Sciences, 3*(4):305–345.

Schneider, W. (1985). Developmental trends in the metamemory-memory behavior relationship: An integrative review. In Forrest-Pressley, D. L., MacKinnon, G. E., & Waller, T. G. (Eds.), *Metacognition, cognition, and human performance: Vol. 1. Theoretical perspectives* (pp. 57–109). Orlando, FL: Academic Press.

Senyk, O., Patil, R. S., & Sonnenberg, F. A. (1989). Systematic knowledge base design for medical diagnosis. *Applied Artificial Intelligence, 3*:249–274.

Shute, V., Glaser, R., & Raghavan, K. (1988). *Inference and discovery in an exploratory laboratory* (Tech. Rep. No. 10). Pittsburgh: University of Pittsburgh, Learning Research and Development Center.

Spohrer, J. C. & Kleiman, R. (1992). Content = knowledge + media: Content acquisition, maintenance, communication, and construction of intelligent tutoring systems. In *Proceedings of the IEEE Conference on Systems, Man, and Cybernetics* (pp. 521–525). Los Alamitos, CA: IEEE Computer Society Press.

Stefik, M. J. (1981). Planning and meta-planning (MOLGEN: Part 2). *Artificial Intelligence, 16*:141–169.

Steier, D. M., Laird, J. E., Newell, A., Rosenbloom, P. S., Flynn, R., Golding, A., Polk, T. A., Shivers, O. G., Unruh, A., & Yost, G. R. (1987). Varieties of learning in SOAR. In Langley, P. (Ed.), *Proceedings of the Fourth International Workshop on Machine Learning* (pp. 300–311). San Mateo, CA: Morgan Kaufmann Publishers, Inc.

Stroulia, E. & Goel, A. (1992). A model-based approach to incremental self-adaptation. In *ML-92 Workshop on Computational Architectures for Supporting Machine Learning and Knowledge Acquisition*, Aberdeen, Scotland.

Sussman, G. J. (1975). *A computer model of skill acquisition.* New York: American Elsevier.

Swanson, H. L. (1990). Influence of metacognitive knowledge and aptitude on problem solving. *Journal of Educational Psychology, 82*:306–314.

Tecuci, G. & Michalski, R. S. (1991). A method for multistrategy task-adaptive learning based on plausible justifications. In *Machine Learning: Proceedings of The Eighth International Workshop* (pp. 549–553). San Mateo, CA: Morgan Kaufmann Publishers, Inc.

van Berkum, J. J. A., Hijne, H., de Jong, T., van Joolingen, W. R., & Njoo, M. (1991). Aspects of computer simulations in an instructional context. *Education and Computing, 6*:231–239.

VanLehn, K. A. & Jones, R. M. (1993). Learning by explaining examples to oneself: A computational model. In Chipman, S. & Meyrowitz, A. L. (Eds.), *Foundations of Knowledge Acquisition: Cognitive Models of Complex Learning* (pp. 25–81). Boston, MA: Kluwer Academic Publishers.

VanLehn, K. A., Jones, R. M., & Chi, M. T. H. (1992). A model of the self-explanation effect. *The Journal of the Learning Sciences, 2*(1):1–59.

Veloso, M. & Carbonell, J. G. (1994). Case-based reasoning in PRODIGY. In Michalski, R. S. & Tecuci, G. (Eds.), *Machine learning IV: A multistrategy approach* (pp. 523–548). San Francisco: Morgan Kaufman Publishers, Inc.

Weinert, F. E. (1987). Introduction and overview: Metacognition and motivation as determinants of effective learning and understanding. In Weinert, F. E. & Kluwe, R. H. (Eds.), *Metacognition, motivation, and understanding*. Hillsdale, NJ: Lawrence Erlbaum Associates, Inc.

Wellman, H. M. (1983). Metamemory revisited. In Chi, M. T. H. (Ed.), *Trends in memory development research.* Basel, Switzerland: S. Karger AG.

Wellman, H. M. & Johnson, C. N. (1979). Understanding of mental process: A developmental study of "remember" and "forget". *Child Development, 50*:79–88.

Wilensky, R. (1984). Meta planning: Representing and using knowledge about planning in problem solving and natural language understanding. *Cognitive Science, 5*:197–234.

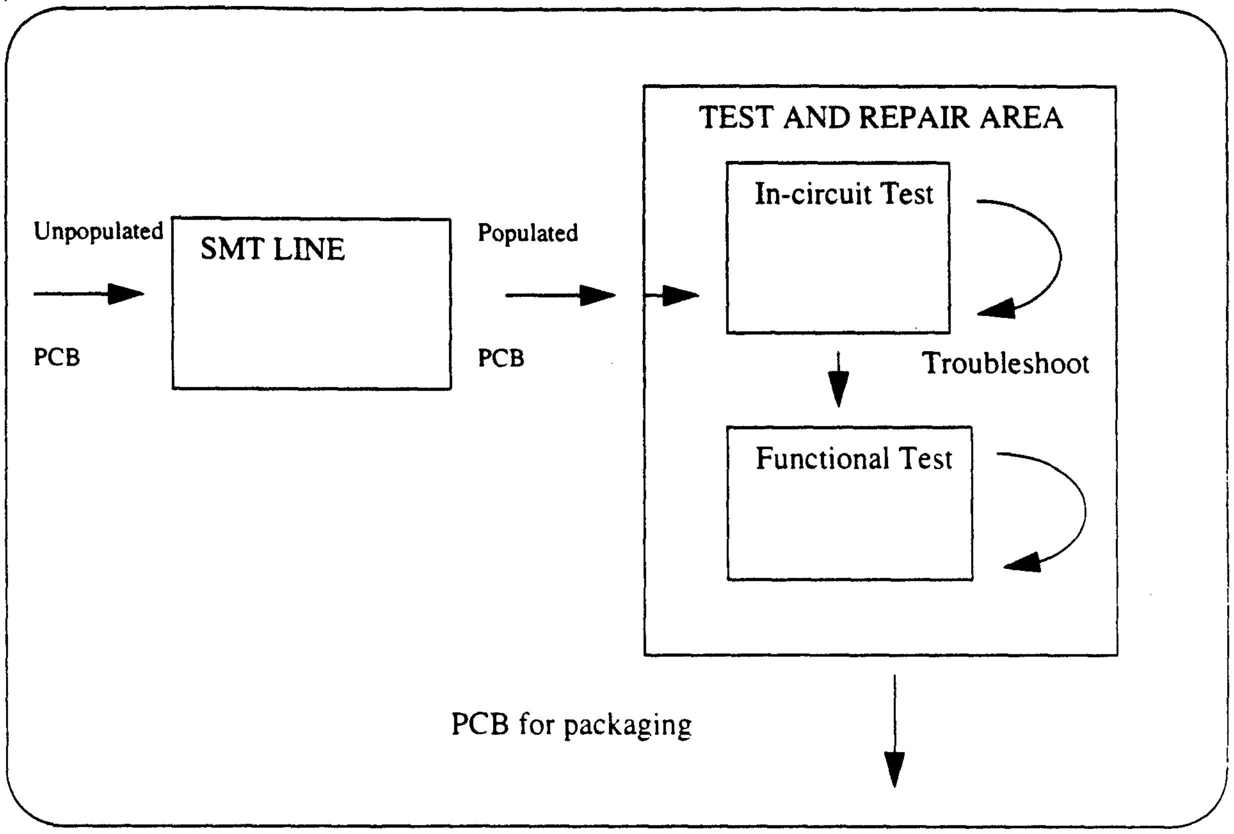


Figure 1: A schematic of the NCR electronics manufacturing plant.

"R24 has a reading slightly lower than nominal. Therefore, the operator suspects that a bad IC connected to it is loading it down. After checking the schematics, he sees that u65 is connected to it. This is a known bad part. He lifts the leg connected to it and ohms out the resistor. The resistor now measures 10K, so he knows that u65 is the culprit. He replaces the IC (and attributes the problem to the vendor)."

Figure 2: Sample troubleshooting protocol from Cohen (1990, p. 206).

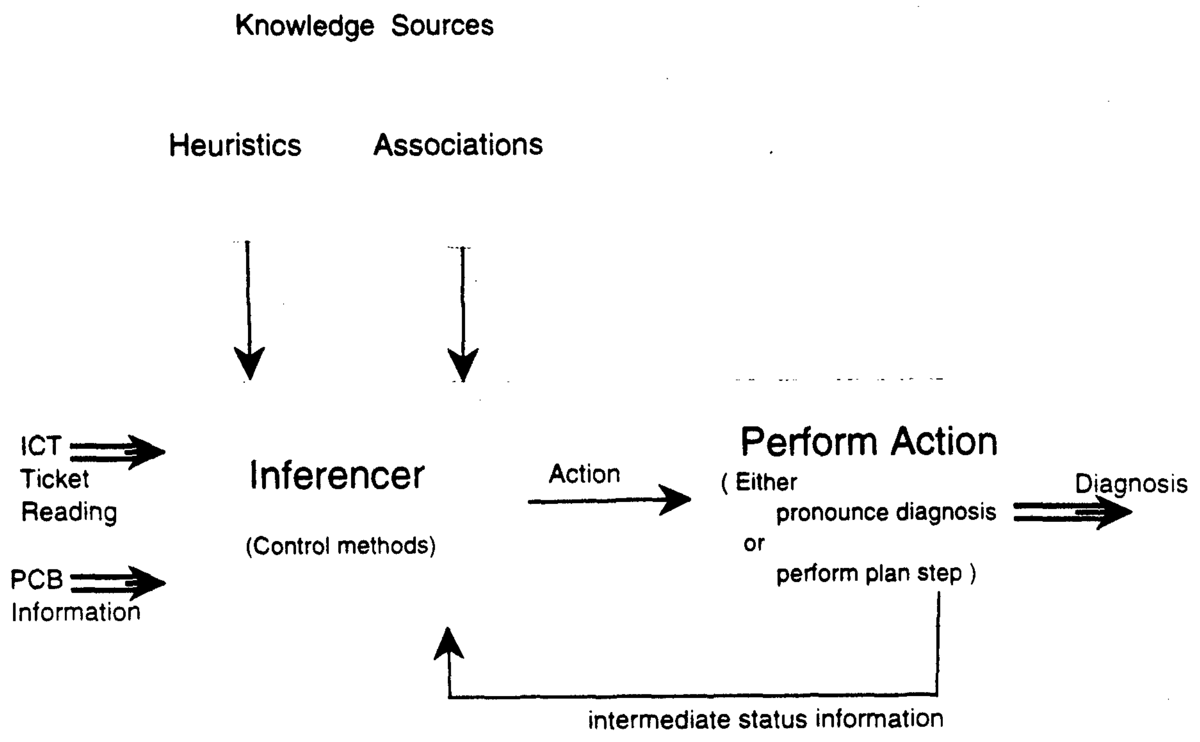
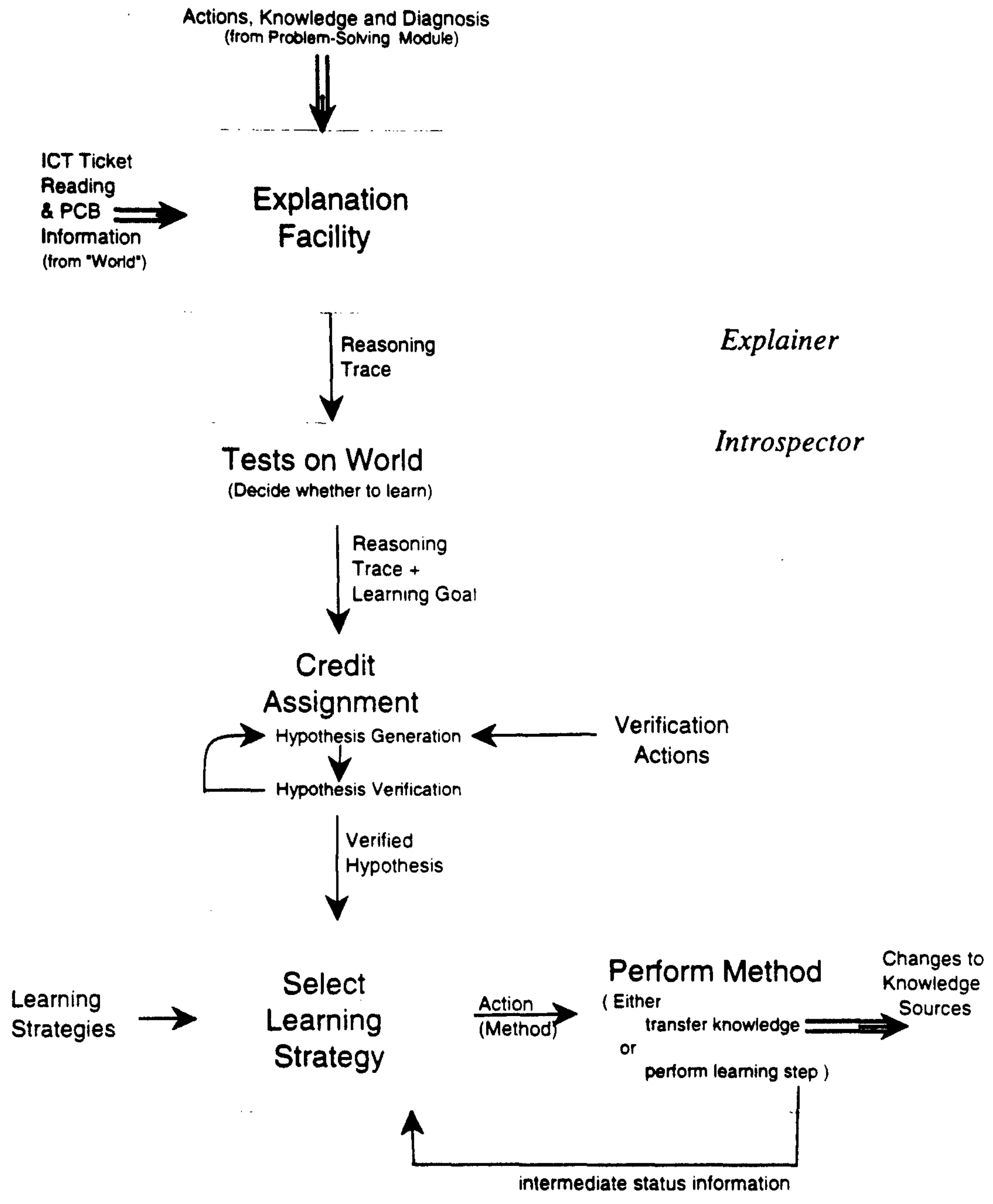


Figure 3: The problem-solving module for the troubleshooting task.

```
Troubleshooting board #6

ICT ticket information
    -------------------------
    These are board faults
    TA 7052 MAX Processor
    -------------------------
    r243 has failed
    Measured = 18.000000 ohms
    Nominal  = 10.000000 ohms
    -------------------------

Entering problem solver

Getting symptom information from ticket
  r243 has failed

Looking for associations for r243
  No associations found

Association search unsuccessful
Diagnosing by heuristics

Looking for heuristics

Applying heuristic-3
  Measured value is much higher than nominal value
  Suspecting an open/defective part

Ohming out on r243
  Ticket reading verified

Performing visual inspection
  Defective part verified

Diagnosis: Defective part, r243 is defective
Repair action: Replace r243
```

Figure 4: An example of a problem-solving episode.



Actions, Knowledge and Diagnosis
(from Problem-Solving Module)
ICT Ticket Reading & PCB Information
(from "World")
Explanation Facility
Reasoning Trace
Explainer
Introspector
Tests on World
(Decide whether to learn)
Reasoning Trace + Learning Goal
Credit Assignment
Hypothesis Generation
Hypothesis Verification
Verification Actions
Verified Hypothesis
Learning Strategies
Select Learning Strategy
Action (Method)
Perform Method
(Either transfer knowledge or perform learning step)
Changes to Knowledge Sources
intermediate status information

Figure 5: Architecture of the multistrategy learning module.

```
// Definition of class Introspector

class Introspector{

 private:

  int tableStrategiesCondition[MAX_PARAMETERS];
                                            // Solution    (1 - correct, 0 - incorrect or no solution)
                                            // Heuristics  (1 - yes, 0 - no)
                                            // Associations (1 - yes, 0 - no)
                                            // Expertinput (1 - yes, 0 - no)
  int qAGoal;                               // True if learning goal requires question-answer session

 public:

    Introspector(void);                     // Default constructor
    TraceCollection traces;                 // Collection of traces for postponement
    void fillTable(void);                   // Method to fill the tableStrategiesCondition
    void executeAppropriateStrategy(void);
                                            // Method to execute appropriate strategy based on table
    void strategyUnH(void);                 // Strategy for completely unsupervised learning
                                            //     when heuristics used for problem solving
    void strategySH(void);                  // Strategy for both supervised and unsupervised learning
                                            //     when heuristics used for problem solving
    void strategySHP(void);                 // strategy for both supervised and unsupervised learning
                                            //     when heuristics used for problem solving
                                            //     and learning goal needs to be suspended
    void strategyDelS(void);                // Strategy to delete obsolete associative knowledge
                                            //     through supervisory input
    void strategyDelUn(char* inputStr);
                                            // Strategy to delete obsolete associative knowledge
                                            //     through unsupervised reasoning
    void learningMethod(void);              // The learning control method
    void reinitializeTable(void);
                                            // Reinitializes tableStrategiesCond
    int solution(void);                     // Determines if solution is correct by performing tests on
                                            //     the world and comparing reasoning trace
    int heuristics(void);                   // Determines if reasoning process involved heuristic knowledge
                                            //     by searching through reasoning trace
    int associations(void);                 // Determines if reasoning trace involved associated knowledge
                                            //     by searching through reasoning trace
    int expertInput(void);                  // Determines if expert input is available
                                            //     by interacting with user
    void setQAGoal (int val)                // Creates learning goal for question-answer session
    int getQAGoal (void)                    // Returns learning goal for question-answer session
    void preQA(char* inString);             // Pre questio-answer steps
    void qA(void);                          // Asks question and gets answer

    void appendTrace (void);                // Utility methods
    void displayTrace (void);
    void displayTrace (char* input);

    void removeTrace (void);
    void removeTrace (char* input);
    Boolean inputTrackTrace(char* input);

};
```

Figure 6: Implementation of introspector as a class in C++.

```
Troubleshooting board #13

ICT ticket information
    -------------------------
    These are board faults
    TA 7052 MAX Processor
    -------------------------
    r22 has failed
    Measured = 16.000000 ohms
    Nominal  = 20.000000 ohms
    -------------------------

Entering problem solver

Step #1
  CONTROL METHOD: get symptom information from ticket
  PRECONDITIONS: ticket available
  MET BY: input
  RESULT: r22 has failed

Step #2
  CONTROL METHOD: find associations for r22
  PRECONDITIONS: symptom available
  MET BY: r22 has failed
  RESULT: no associations for r22

Step #3
  CONTROL METHOD: find heuristics
  PRECONDITIONS: symptom available AND no associations
  MET BY: r22 has failed AND no associations for r22
  RESULT: heuristic-1 found

Step #4
  HEURISTIC: apply heuristic-1
  PRECONDITIONS: measured value is slightly lower than nominal value
  MET BY: ticket information
  RESULT: ohming-out recommended

Step #5
  ACTION: ohming-out on r22
  PRECONDITIONS: symptom available AND action recommended
  MET BY: r22 has failed and ohming-out recommended
  RESULT: bogus ict ticket

Diagnosis: bogus ict ticket

Entering learner

Analyzing Trace Meta-XP steps 1-5
  SYMPTOM: r22 has failed
  DIAGNOSIS: bogus ict ticket
  OUTCOME: diagnosis correct
Using Introspective Meta-XP
  GOAL: learn association for r22
  STRATEGY: unsupervised learning of associations

Invoking learning strategy
Creating new association for r22

Meta-TS now has associations for
r243, u37, r254, r200, r121, u73, r22
```

Figure 7: An example of the strategy of unsupervised learning of associations in Meta-TS. Here, Meta-TS has just learned the association that r22 failures often indicate a "bogus" ticket reading.

```
Troubleshooting board #2

ICT ticket information
    -------------------------
    These are board faults
    TA 7052 MAX Processor
    -------------------------
    r24 has failed
    Measured = 21.200001 ohms
    Nominal  = 10.000000 ohms
    -------------------------

Entering problem solver

Step #1
  CONTROL METHOD: get symptom information from ticket
  PRECONDITIONS: ticket available
  MET BY: input
  RESULT: r24 has failed

Step #2
  CONTROL METHOD: find associations for r24
  PRECONDITIONS: symptom available
  MET BY: r24 has failed
  RESULT: no associations for r24

Step #3
  CONTROL METHOD: find heuristics
  PRECONDITIONS: symptom available AND no associations
  MET BY: r24 has failed AND no associations for r24
  RESULT: heuristic-3 found

Step #4
  HEURISTIC: apply heuristic-3
  PRECONDITIONS: measured value is much higher than nominal value
  MET BY: ticket information
  RESULT: ohming-out and visual-inspection recommended

Step #5
  ACTION: ohming-out on r24
  PRECONDITIONS: symptom available AND action recommended
  MET BY: r24 has failed and ohming-out recommended
  RESULT: ticket reading verified

Step #6
  ACTION: visual-inspection on r24
  PRECONDITIONS: symptom available AND action recommended
  MET BY: r24 has failed and visual-inspection recommended
  RESULT: r24 is missing

Diagnosis: missing part, r24 is missing

Entering learner

Analyzing Trace Meta-XP steps 1-6
  SYMPTOM: r24 has failed
  DIAGNOSIS: missing part, r24 is missing
  OUTCOME: diagnosis incorrect, u37 is defective
Using Introspective Meta-XP
  GOAL: learn association for u37
  STRATEGY: supervised learning of associations

Invoking learning strategy

Is expert input available for this episode?  >> yes
  Enter left-hand-side of association:  >> u37
  Select right-hand-side of association:
  a. Bogus ICT ticket
  b. Replace defective part
  c. Perform visual inspection followed by diagnosis
  d. Perform lifted leg procedure followed by diagnosis
  >> c

Creating new association for u37

Meta-TS now has associations for
r243, u37
```

Figure 8: An example of the strategy of supervised learning of associations in Meta-TS. Italics indicate user input during this episode.

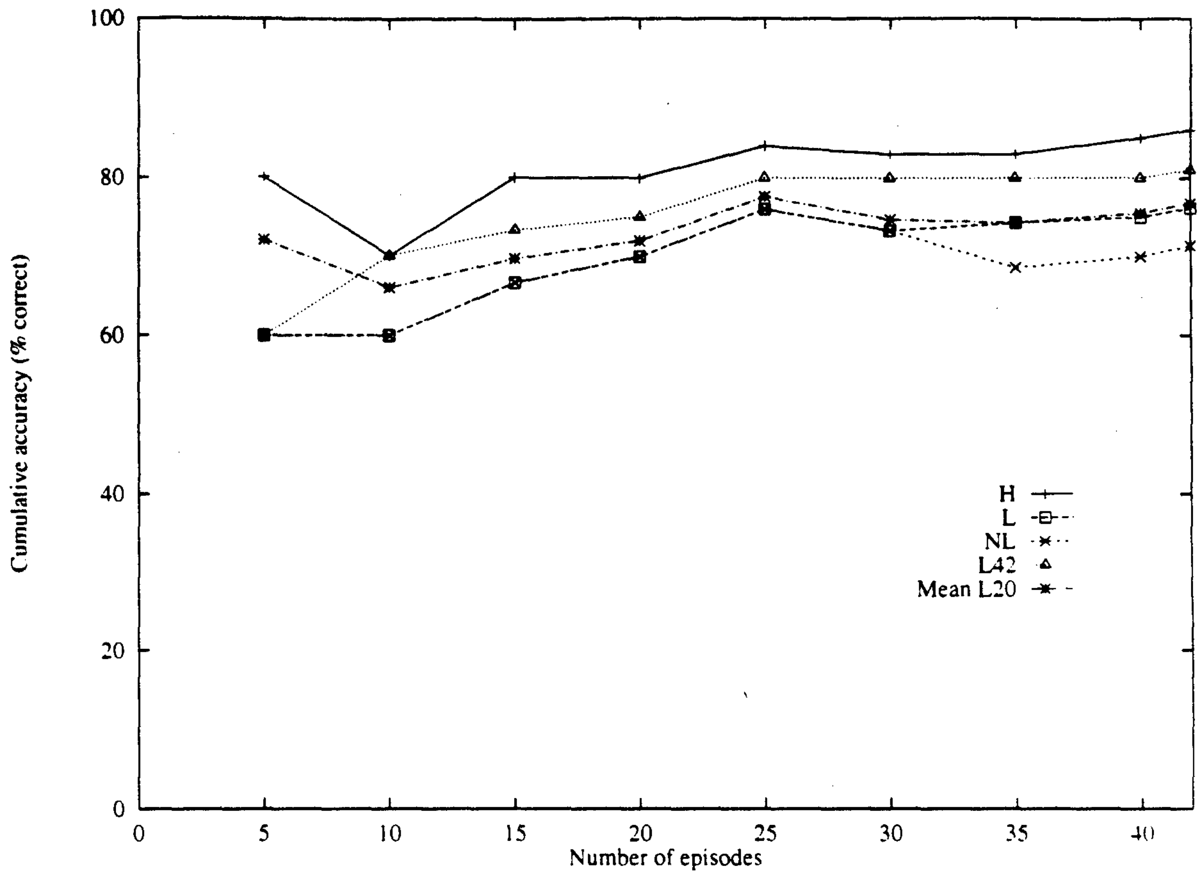


Figure 9: Cumulative diagnostic accuracy

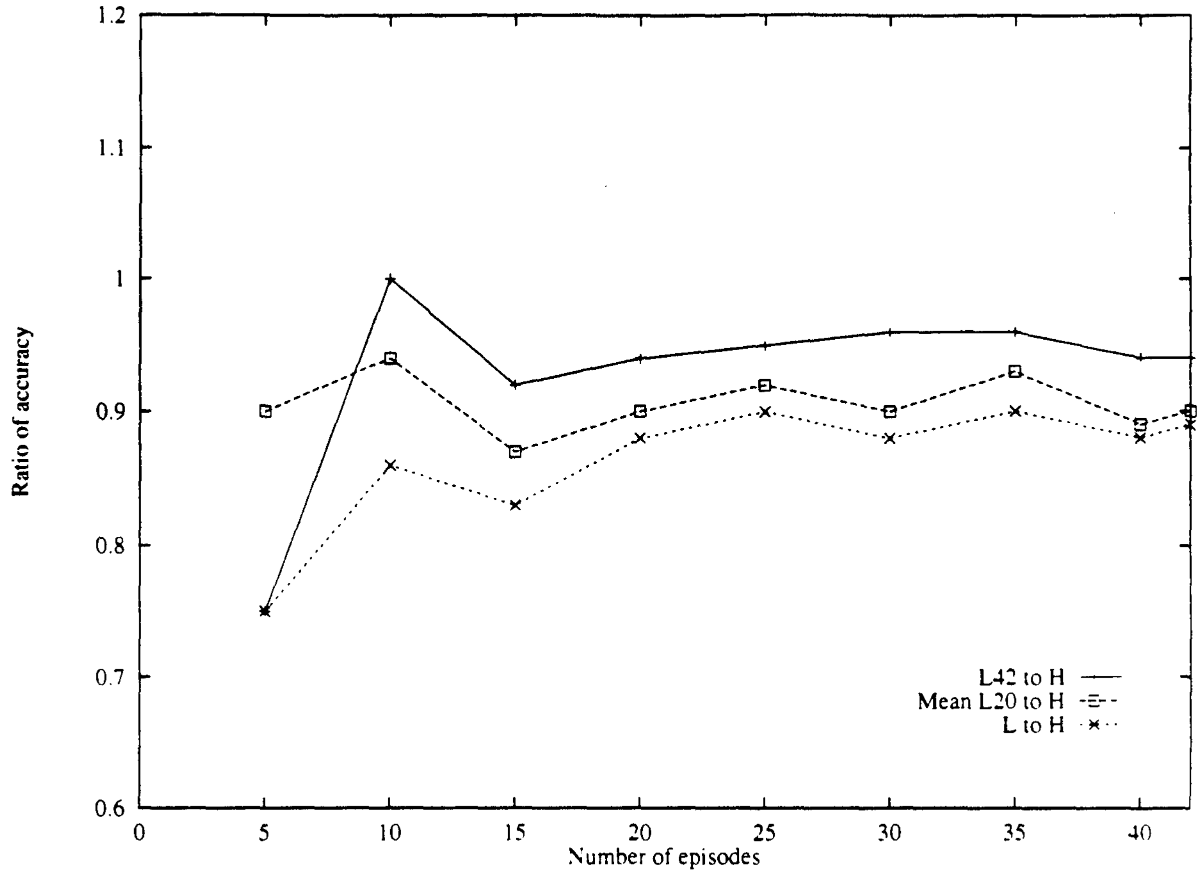


Figure 10: Ratio of learning conditions to hand-coded condition in terms of diagnostic accuracy

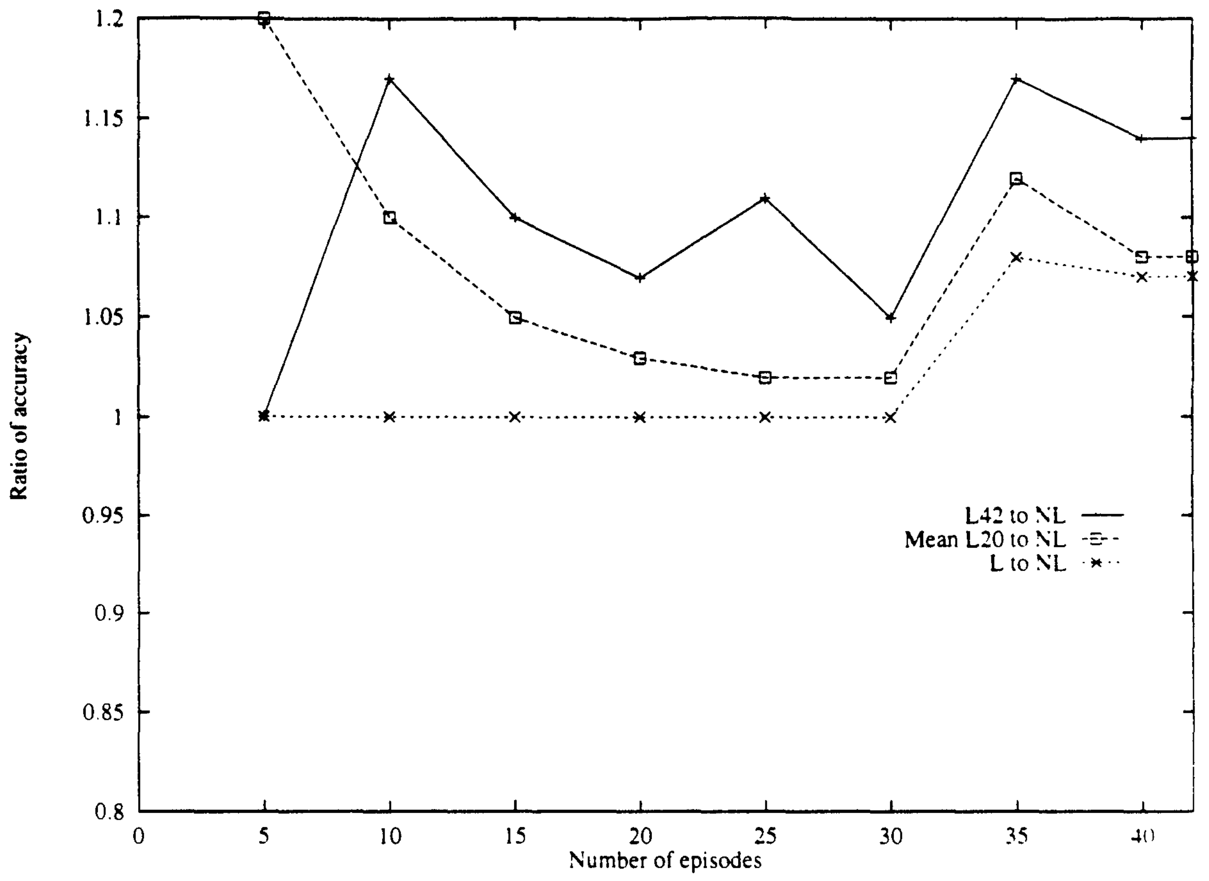


Figure 11: Ratio of learning conditions to non-learning condition in terms of diagnostic accuracy

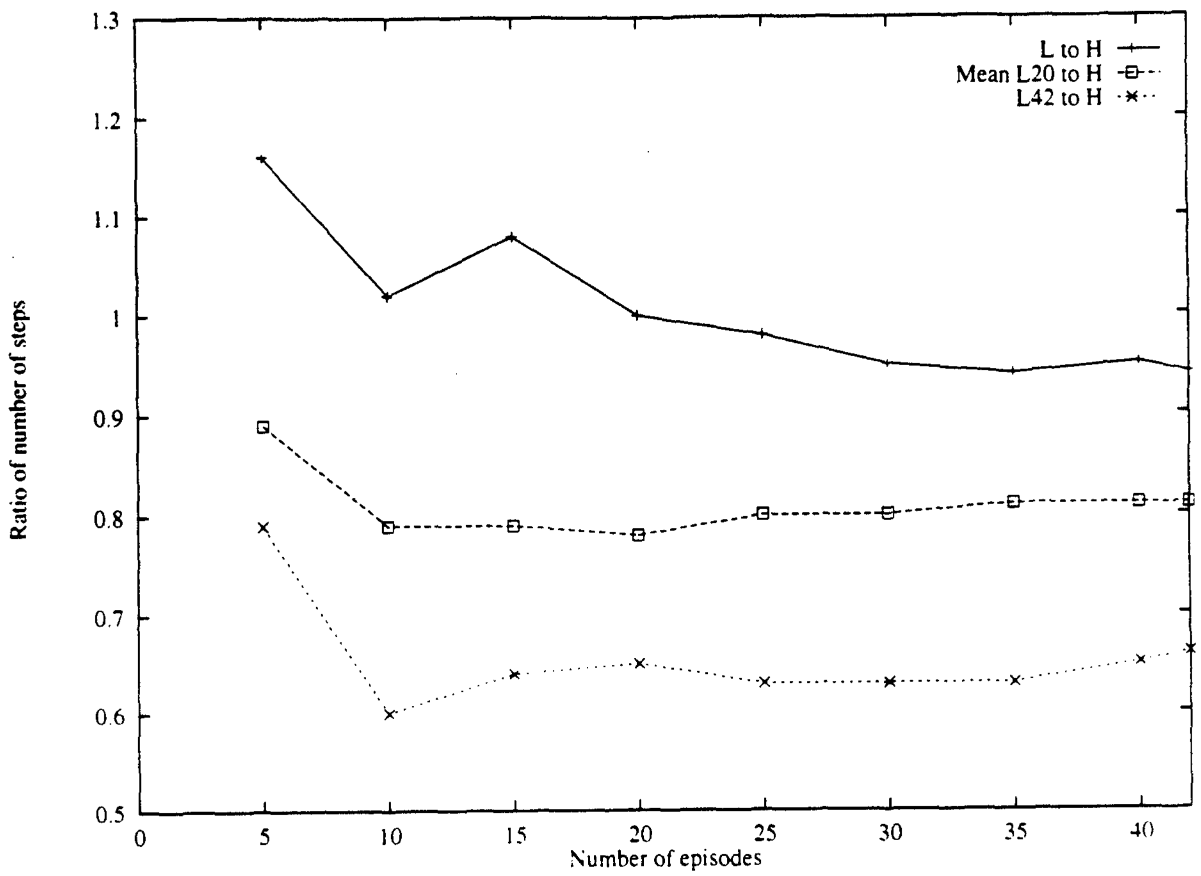


Figure 12: Ratio of hand-coded condition to learning conditions in terms of number of steps to solve problem

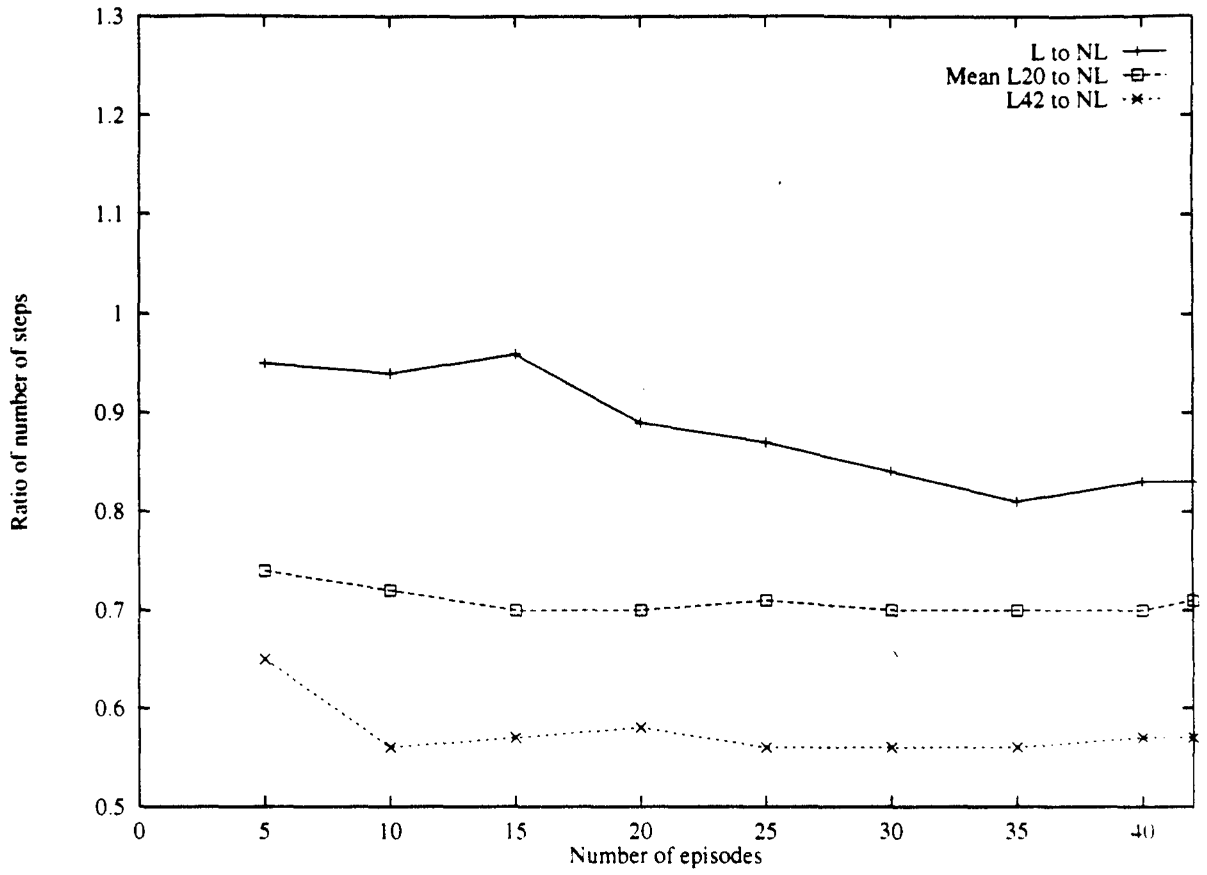


Figure 13: Ratio of non-learning condition to learning conditions in terms of number of steps to solve problem

Tables

Table 1: Algorithm for introspective multistrategy learning in Meta-TS. Note that step 2E is not necessarily performed immediately after 2D; in some cases, it may be performed at a later time (for example, as in the case of the postponement strategy in which learning is deferred until a suitable opportunity arises).

Step 0: Perform troubleshooting and record in Trace Meta-XP, including reasoning steps and knowledge (associations or heuristics) used in each step.

Step 1: Analyze Trace Meta-XP to identify reasoning failures, including incorrect diagnosis, inability to create a diagnosis, and correct diagnosis but through inefficient problem-solving.

Step 2: If analysis reveals a reasoning failure, then learn:

- **Step 2A:** Characterize type of reasoning failure
- **Step 2B:** Use Introspective Meta-XPs encoded in introspector to determine cause of failure
- **Step 2C:** Use analysis of type and cause of failure to determine what to learn
- **Step 2D:** Choose appropriate learning algorithm
- **Step 2E:** Apply learning algorithm

Table 2: Examples of associative and heuristic knowledge used in the problem-solving module. r# indicates the number of a resistor component, and u# indicates the number of an IC (integrated circuit) component.

Associative knowledge
• r254 is often damaged. Visually inspect the part. If it is damaged, replace the part.
• If r1 or r2 fails, the ticket reading is "bogus."[a] Output the diagnosis "Bogus ICT ticket."
• u56 and u65 are known bad parts. Use the "lifted leg" procedure[b] to identify the bad part(s) and replace them.
• r228, r239 and r279 are connected to u51. If one of these has failed with a low reading, u51 should be replaced.

Heuristic knowledge
• If the measured reading of a resistor is slightly higher than the nominal value on the ICT ticket, perform the "visual inspection" procedure. If the defect is found, terminate the search, otherwise output the diagnosis "Unable to make an inference" and perform appropriate repair action.
• If the measured reading of the resistor is slightly lower (qualitatively) than the nominal value, perform the "ohming out" action.[c] If the diagnosis is "Bogus ICT ticket", terminate the search, otherwise perform the "check schematics"action[d] and make an ordered list of faulty ICs. If any of these can be fixed by association-based search, terminate search, otherwise test each of these ICs to determine the faulty component. If a defective component is not found, terminate the search and output the diagnosis "Unable to make an inference."

[a] A "bogus" ICT ticket reading is typically caused when there is a poor connection between the board and the tester.

[b] During the "lifted leg" procedure the operator uses a dental tool to tug at each leg on a component to find legs which have not been soldered to the pad.

[c] "Ohming out" refers to using a multimeter to check the resistance of a connection on the board. This procedure involves touching the two probes on the multimeter to each end of the connection.

[d] "Check schematics" refers to the procedure followed by an operator to find the list of parts connected to a particular component (using the schematic page number for parts provided by the ICT ticket).

Table 3: Examples of troubleshooting actions, control methods, and repair actions in the problem-solving module.

Troubleshooting actions

- Perform visual inspection.
- Check for faulty IC that lowers resistance.
- Ohm out on a resistor.
- Check schematics.

Control methods

- Look at the symptom information on the ICT ticket first.
- Determine if there is an association for that symptom in memory; if so, invoke it and terminate the search.
- Perform the "visual inspection" action. If the defect is found, suggest appropriate repair action and terminate search. Use the appropriate heuristics and determine the repair action depending on the qualitative difference between the measured and nominal reading in the ticket.

Repair actions

- Identify the part number of the defective part and replace it with an equivalent part.
- Output "Bogus ICT ticket reading" to indicate a suspected false ICT ticket reading.
- Identify the part number of the missing part and install an appropriate part.
- Output "Unable to make an inference" to indicate insufficient knowledge to arrive at an inference that indicates a repair action.

Chapter 1

Learning, Goals, and Learning Goals

Ashwin Ram and David B. Leake

1 Why goals?

In cognitive science, artificial intelligence, psychology, and education, a growing body of research supports the view that learning is largely a goal-directed process. Experimental studies show that people with different goals process information differently; work in machine learning presents functional arguments for goal-based focusing of learner effort. Recent work in these fields has focussed on issues of how learning goals arise, how they affect learner decisions of when and what to learn, and how they guide the learning process. It is increasingly evident that investigation of goal-driven learning can benefit from bringing these perspectives together in a multidisciplinary effort (Leake & Ram, 1993).

The central idea underlying goal-driven learning is that, because the value of learning depends on how well the learning contributes to achieving the learner's goals, the learning process should be guided by reasoning about the information that is needed to serve those goals. The effectiveness of goal-driven learning depends on being able to make good decisions about when and what to learn, on selecting appropriate strategies for achieving the desired learning, and on guiding the application of the chosen strategies. Research into such topics includes the development of computational models for goal-driven learning, the testing of those models through psychological experiments and empirical experiments with computer programs, the justification of the models through functional arguments about the role and utility of goals in learning, and the use of models of goal-driven learning in guiding the design of educational environments. The common themes in these research efforts are the investigation of types of learning goals, the origins of learning goals, and the role of goals in the learning process.

Research on goal-driven learning in artificial intelligence has been motivated largely by computational arguments. The problem of combinatorial explosion of inferences is well known; in any realistic task domain, time and resource constraints prohibit consideration of all but a few of the possible inferential paths. Consequently, any reasoner, human or machine, must focus its attention and resources on pursuing those inferential paths that are likely to be most useful. Similarly, in any realistic situation, there are several different types of learning that a reasoner might perform, several kinds of new knowledge that a reasoner might acquire, and several kinds of reformulation or reorganization of existing knowledge that a reasoner might carry out. Again, due to time and resource constraints it is only practical to perform a few of these operations. Consequently, the reasoner must focus its attention and resources on executing the learning operations that are likely to be most useful. Because the utility of an inference or a piece of knowledge can best be evaluated relative to a particular task or goal, goal-based considerations must guide reasoning and learning.

In addition to these computational arguments for goal-driven learning, research in goal-driven learning has a cognitive basis in psychological research. This research has established much evidence for the influence of goals and beliefs on human learning, and for the use of active, strategic, and goal-driven processes in many kinds of learning that humans perform. However, many questions remain concerning the kinds of goals that people pursue, the conditions under which those goals influence learning, and the kinds of learning that are influenced by those goals.

Research in cognitive science combines the cognitive perspective of psychology with the computational perspective of artificial intelligence, developing computational models of human learning that are evaluated using computational metrics as well as by comparison with human performance. Research in education has also been concerned with psychological data about human learning, but from a pragmatic perspective. This research has attempted to use empirical evidence to guide the design of instructional and educational scenarios so as to facilitate learning, taking as its starting point the evidence for facilitation of certain kinds of learning by particular kinds of goals. These scenarios have also been used as the basis for further psychological experimentation to validate the underlying theories. In this chapter, we describe a framework for goal-driven learning and its relationship to prior and current theories from each of these perspectives.

2 An everyday example

Goal-driven learning is triggered when a reasoner needs to learn in order to improve its performance at some task. A goal-driven learner determines *what* to learn by reasoning about the information it needs, and determines *how* to learn by reasoning about the relative merit of alternative learning strategies in the current circumstances. For example, for a first-time stereo buyer, the goal of getting good buy on a stereo may give rise to at least two learning goals: a goal to learn the best sources for sound equipment and to a goal to learn how to judge the merits of competing equipment. Each of these learning goals may trigger learning subgoals. In order to learn the best place to buy sound equipment, the buyer may first have to learn general criteria for what constitutes a good store for buying sound equipment, and then specifics about prices, service, etc. to classify different stores. In order to learn how to judge particular equipment, the buyer will have to learn about the classes of alternatives available and about specific equipment within those classes. Thus some learning goals involve gathering information in the external world, while others involve reformulating or changing information that is already known, by operations such as forming generalizations or reorganizing memory.

In order to perform the desired learning, the stereo buyer must select strategies for accomplishing each of its learning goals. For example, the buyer may choose between learning strategies including asking others' opinions, reading magazine articles, forming inductive or explanation-based generalizations from demonstrations of equipment, or even disassembling equipment to determine the quality of its electronic components. Learning strategy selection depends on factors such as the buyer's prior knowledge, the buyer's resources (e.g., how much time the buyer can spend on the shopping process), opportunities (e.g., happening to meet an expert on sound equipment at a party), and the buyer's own abilities (e.g., whether the buyer has the expertise to judge the quality of equipment by disassembling it).

This example illustrates the value of goal-driven learning in focusing learner effort, and also suggests the range of roles that goals can play in influencing learning. Goals determine how much effort to allocate to performance tasks (e.g., the task of buying a stereo), indirectly influencing the resources available for the learning that will be performed as part of that task. Goals also determine the focus of attention when new information is received as input (e.g., focusing attention on announcements of stereo sales). They determine what should be learned (e.g., determining that it is worthwhile to generalize about relationships between store types and prices). They give criteria for evaluating the results of learning and deciding what learned information to store (in this example, the value of learning is its usefulness for guiding the shopping decision). Table 1 summarizes these and other possible roles of goals in learning. In the following sections we concentrate on developing a framework and terminology on which to base our analysis of goal-driven learning, and after developing that framework we return to the ways that goals affect learning in section 8.

3 Towards a planful model of learning

As the previous example illustrates, a goal-driven learner makes decisions about what, how, and when to learn in order to further its goals. In this view, learning can be considered a "planful" process (e.g., Etzioni, Hanks, Weld, Draper, Lesh, & Williamson, 1992; Hunter, 1990/chapter 2; Leake/chapter 20; Michalski &

Guiding the performance task by:

- Determining the resources made available to the performance task
- Guiding the control or search procedure used in the performance task
- Guiding retrieval of plans, problem solutions, and other types of knowledge
- Focusing attention on certain aspects of the input
- Guiding the evaluation of the outcome of the performance task

Guiding the learning task, by:

- Specifying the target of learning (desired output of a learning algorithm)
- Selecting the learning algorithms to be used
- Constraining the learning process (for example, influencing the policies under which the learning algorithms operate)
- Focussing the search for information needed to carry out the learning
- Determining when learning should be attempted
- Aiding evaluation of results of learning with respect to the desired output

Guiding storage, by:

- Selecting what to store
- Determining how learned knowledge is indexed

Table 1: Ways in which goals can influence learning.

Ram, chapter 21; Pryor & Collins, 1992/chapter 10; Ram & Cox, 1994/chapter 7; Ram, Cox, & Narayanan, chapter 18; Ram & Hunter, 1992/chapter 4; Redmond, 1992; Quilici, in press; Schank & Abelson, 1977; Xia and Yeung, 1988/chapter 12). This learning process is analogous to models of problem solving in which the reasoner uses task goals to formulate action plans for achieving these goals (e.g., Newell & Simon, 1972; Greeno & Simon, 1988; VanLehn, 1989). Learning actions or schemas are selected, combined, and invoked appropriately on the basis of existing learning goals and available environmental opportunities for learning. Learning is a behavior explicitly carried out to seek information, driven by needs arising from the reasoner's performance on a task that learning is intended to facilitate, and mediated by the formulation and manipulation of explicit learning goals.

The motivation for the goal-driven approach is to control processing in a rich world. Simply put, knowledge that is valid in principle need not necessarily be useful (Mitchell & Keller, 1983); thus, it is desirable to avoid the effort involved in learning knowledge that does not contribute to the reasoner's overall purpose. For example, Ram and Hunter (1992/chapter 4) argue that, due to the computational complexity of reasoning about the combinatorially large number of inferences that are possible in any realistic situation, it is essential to focus inferential and learner effort on deriving those pieces of knowledge that are likely to be most useful. Hunter (1990/chapter 2) argues that inference during learning (such as inductive inference) is also potentially combinatorially explosive and that explicit consideration of desirable knowledge should be used to guide this inference. Likewise, Leake (1992) argues for similar reasons that decisions about what to learn about new situations must be driven by characterizations of the learner's information needs. Theoretical analyses (desJardins, 1992/chapter 8; Etzioni, 1992; Francis & Ram, 1993; Gratch & DeJong, 1993), as well as empirical investigations of the utility of learning (Minton, 1990/chapter 3; Tambe, Newell, & Rosenbloom, 1990) provide support for this argument. Active, goal-driven learning implies the ability to make explicit decisions about what, when, and how to learn (Ram, Cox, & Narayanan/chapter 18). Thus some of the motivations for goal-based approaches include (see also Cox & Ram, 1994):

- **Alleviating problems of computational complexity:** The ability of a reasoner to make decisions about its reasoning and learning processes helps to alleviate problems caused by the computational complexity of reasoning in an open world, by enabling the reasoner to focus its efforts towards processing that serves its goals (Cox, 1993; Hunter, 1990/chapter 2; Leake, 1992; Leake/chapter 20; Ram & Hunter, 1992/chapter 4). An analysis of the utility of learning can help in determining the target of learning (desJardins, 1992/chapter 8), in guiding learning processes (Gratch & DeJong, 1993; Gratch, DeJong, & Chien, 1994; Provost, 1994), and also in deciding whether to learn at all (Markovitch & Scott, 1993; Minton, 1990/chapter 3).

- **Facilitating the use of opportunities to learn:** If a reasoner does not have sufficient resources at the time it realizes it has a need to learn, or if the requisite knowledge is not available at that time, the reasoner can suspend its learning goals in memory so that they can be retrieved and pursued at a later time (Hunter, 1990/chapter 2; Hammond, Converse, Marks, & Seifert, 1993; Ram, 1991, 1993; Ram, Cox, & Narayanan/chapter 18; Ram & Hunter, 1992/chapter 4).

- **Improving the global effectiveness of learning:** Taking goal priorities and goal dependencies into account when deciding what to learn and how to coordinate multiple learning strategies improves the effectiveness of learning in a system with multiple goals. Learning strategies, represented as methods for achieving learning goals, can be chained, composed, and optimized, resulting in learning plans that are created dynamically and pursued in a flexible manner (Cox, 1993; Cox & Ram, 1994; Gratch, DeJong, & Chien, 1994; Hadzikadic & Yun, 1988; Hunter, 1990/chapter 2; Michalski, 1993; Michalski & Ram, chapter 21; Ram & Hunter, 1992/chapter 4; Redmond, 1992; Stroulia & Goel, 1994).

- **Increasing the flexibility of learning:** In situations involving multiple reasoning failures, multiple active and suspended learning goals, multiple applicable learning strategies, and limited resources, direct mapping from specific types of failures to individual learning strategies is impossible, and an active, planful approach becomes necessary. For a given failure, there may be more than one algorithm which needs to be applied for successful learning and, conversely, a given algorithm may apply to many different types of failures (Cox, 1993; Cox & Ram, 1994; Krulwich, Birnbaum, & Collins, 1993; Ram, Cox, & Narayanan/chapter 18). A planful model of learning allows decoupling of many-to-many relationships, leading to more flexible behavior (Cox, 1993, Cox & Ram, 1994).

- **Improving management of interactions between learning processes:** Explicit formulation of learning goals facilitates detection of dependency relationships, so that goal violations can be avoided (Cox, 1993, Cox & Ram, 1994). When multiple items are learned from a single episode, the changes resulting from one learning algorithm may affect the knowledge structures used by another algorithm. Such dependencies destroy any implicit assumption of independence built into a given learning algorithm that is used in isolation. For example, one learning algorithm may split a concept definition into separate schemas or otherwise modify the definition. Therefore, an indexing algorithm that uses the attributes of concepts to create indices must necessarily follow the execution of any algorithm that changes the conceptual definition.

Psychological evidence also supports the existence of goal-based influences on human focus of attention, inference, and learning (e.g., Barsalou, 1991/chapter 5; Faries & Reiser, 1988 Hoffman, Mischel, & Mazze, 1981; Ng & Bereiter, 1991/chapter 14; Seifert 1988, Srull & Wyer, 1986; Wisniewski & Medin, 1991/chapter 6; Zukier, 1986; see also discussion by Hunter, 1990/chapter 2). These ideas are related to the "goal satisfaction principle" of Hayes-Roth and Lesser (1976), which states that more processing should be given to knowledge sources whose responses are most likely to satisfy processing goals, and to the "relevance principle" of Sperber and Wilson (1986), which states that humans pay attention only to information that seems relevant to them. Those principles make sense because cognitive processes are geared to achieving a large cognitive effect for a small effort. To achieve this, the understander must focus its attention on what seems to it to be the most relevant information available. Goals can facilitate learning even when they are not generated internally by the reasoner; for example, Steinbart (1992) shows that asking users questions (i.e., "creating" knowledge goals in people) can help them learn from a computer-assisted training program, and Patalano, Seifert, and Hammond (1993) show that presenting users with a goal and a plan to achieve it can facilitate later detection of relevant features of a situation. There is also much research on the origins of goals; for example, Graesser, Person, and Huber (1992) discuss several types of questions, or goals to seek information, and the cognitive mechanisms that generate them. Many of these are related to the learning goal formulation mechanisms discussed here.

The goal-driven learning framework does not imply that all processing is explicitly goal-driven. A reasoner that was completely goal-driven would only notice what it was looking for already; it would not be able to respond to and learn from unexpected input. Instead, it is reasonable to assume that there would be some automatic, bottom-up, or non-goal-driven processing during reasoning and learning, which would support strategic, top-down, or goal-driven processes such as those discussed here (e.g., Barsalou/chapter 17; Kintsch, 1988; Leake, 1992; McKoon & Ratcliff, 1992; Ram, 1991).

A significant body of psychological research points to the influence of "metacognition"—cognition by a person concerning that person's own cognitive processes—in human performance (e.g., Forrest-Pressley, MacKinnon & Waller, 1985; Weinert, 1987; Wellman, 1985, 1992). Gavelek and Raphael (1985) discuss a form of metacognition, called metacomprehension, which addresses the abilities of individuals to adjust their cognitive activity in order to promote more effective comprehension, in particular, the manner in which questions generated by sources external to the learner (i.e., from the teacher or text), as well as those questions generated by the learners themselves, serve to promote their comprehension of text. White and Gunstone (1989) argue that resolution of conflicting beliefs and permanent conceptual change requires "metalearning"—control over one's learning. For example, they discuss a study by Gauld (1986) that shows that students who learn new scientific beliefs often revert to their original beliefs over time because they have merely accepted the new knowledge without any real commitment to it. They argue that deep reflection on one's beliefs is a key part of the awareness and control over one's learning, and suggest methods for promoting metalearning in science classrooms.

It is clear, of course, that humans cannot exert explicit meta-control over all their learning processes, and the level of control that can be exerted, as well as how it is exerted, remain open questions. It is also possible (though, in our opinion, unlikely) that it may turn out not to be efficient to use this framework as a technological basis for the design of computer programs that learn. Nevertheless, the framework presented here may be used to take an intentional stance (Dennett, 1987) towards a reasoner for the purposes of building a computational model of learning. In such a stance, the competence of the reasoner can be modelled using goals, learning decisions, learning actions, and so forth as the basic theoretical constructs in task-level and algorithm-level descriptions of the reasoner. That stance can be taken without any commitment to existence of these constructs at the implementational level of, say, neural representations and processes in the human brain, or to the degree of conscious self-awareness of these processes in human thought.

4 A framework for goal-driven learning

In order to form a unified view of the diverse research results on goal-driven learning, we propose a general framework that describes the goal-driven learning process. While no single piece of research to date has investigated this framework as a whole or exactly as stated, the framework serves to provide an integrative structure into which individual research efforts fit as pieces of the puzzle of goal-driven learning. The key idea behind our framework is to model learning as an *active* (explicitly goal-driven) and *strategic* (rational and deliberative) process in which a *reasoner*, human or machine, explicitly identifies its goals in learning and attempts to learn by determining and pursuing appropriate learning actions via explicit reasoning about its goals, its abilities, and environmental opportunities.

In this framework, learning is motivated by the *performance tasks* that the reasoner is attempting to perform in the world. The performance tasks give rise to *task goals*, as well as subgoals of those goals, and subtasks to achieve them. As the tasks and subtasks are performed, the reasoner formulates explicit *learning goals* to perform types of learning which, if successful, would improve its ability to carry out those performance tasks or subtasks. The learning goals, in turn, guide the learning behavior of the reasoner, leading it to focus attention, allocate resources, and select appropriate *learning algorithms* or *learning strategies* when opportunities to learn arise. In our previous example, the top-level task goal would be to get a good buy on a stereo, which would spawn subtasks such as going to a store and purchasing the stereo. These subtasks give rise to learning goals to learn information needed to select the store and the stereo to buy. Some

Reasoner	An intelligent system, human or machine.
Performance task(s)	Overall task(s) that the reasoner is performing that create an effect on the external world.
Task goals	Specific goals and subgoals to be accomplished in order to accomplish the performance task.
Learning goals	Goals to learn (including learning by acquiring knowledge, reorganizing or reformulating knowledge, verifying hypotheses, etc.). These include both a description of the needed information and information about the task for which the information is needed.
Strategies or methods	Processing steps that accomplish a goal.
Algorithms	Computational formulations of strategies.
Reasoning trace	Record including information on goal-subgoal decomposition of goals, choice of methods to accomplish goals and subgoals, and other decisions taken in pursuing those goals, as well as the bases for these decisions, results of reasoning actions, alternative courses of action, etc.

Table 2: Summary of terminology.

of those learning goals may seek to gather information about the external world, while others may seek to create generalizations, test hypotheses, reorganize memory, or otherwise change existing knowledge. Those learning goals prompt the choice of learning strategies such as "shopping around," looking at reviews in magazines, and so forth.

The goal-driven learning process involves not only learning about the world, but also learning to improve the reasoner's own reasoning process. In order to identify the learning that needs to occur, the reasoner needs to be able to analyze its reasoning process in addition to the knowledge that the reasoner invokes during the reasoning process. To facilitate this, the reasoner maintains a *reasoning trace* of its internal decision-making. The reasoning trace provides the basis for *introspective reasoning* or *meta-reasoning* to guide learning and improve its reasoning performance. Table 2 summarizes the terminology we will use in our framework.

More concretely, goal-driven learning can be modeled as a two-step process. The first step involves the generation of learning goals based on the performance tasks and task goals of the reasoner. This step can be thought of as the process of deciding what to learn, and results in the formulation of learning goals that specify the desired learning that is to occur as well as the origin of the need for this learning. The second step involves the pursuit of learning goals based on the reasoner's needs, its resources, and on environmental factors that determine the timeliness of pursuing certain learning actions in a given situation. This step can be thought of as the process of deciding how and when to learn and carrying out the learning. When the learning actually occurs, this step results in the satisfaction of one or more of the reasoner's learning goals.

Step 1: Generating learning goals: Figure 1 describes the process by which learning goals are generated. The reasoner is assumed to be pursuing a performance task that can be characterized in terms of the current situation and task goals specifying the desired result of the task. In the stereo example, the situation might be that the shopper lives in New York, knows nothing about stereos, and has $500 to spend; the task goal would be to buy a stereo that was a good value for the $500 price range.

Given the performance task, the reasoner performs reasoning in support of that task and maintains a trace reflecting its reasoning process. The reasoning trace records the goal-subgoal decompositional structure of the task goals, the choice of methods for achieving them and other decisions taken, the factors influencing those decisions, and descriptions of other reasoning actions (e.g., attempts to retrieve information) and their outcomes (Carbonell, 1986; Ram & Cox, 1994/chapter 7). For example, forming an executable plan to get a good buy on a stereo requires knowing which stereo to buy and where to buy it. If the reasoner does not know, a reasoning failure occurs because current knowledge is insufficient to make a decision.

At a suitable point in processing, the reasoning trace and its results are evaluated in light of the reasoner's task goals. If any problems arose during processing, learning is needed to enable the reasoner to avoid similar problems in the future. In being driven by deficiencies in the reasoner's knowledge, the process for generating learning goals is in the spirit of impasse-driven or failure-driven learning (e.g., Chien, 1989; Collins & Birnbaum, 1988; Hammond, 1989; Kocabas, 1994; Laird, Newell, & Rosenbloom, 1986; Mooney & Ourston, 1993; Mostow & Bhatnagar, 1987; Newell, 1990; Owens, 1991; Park & Wilkins, 1990; Ram & Cox, 1994/chapter 7; Riesbeck, 1981; Schank, 1982; Schank & Leake, 1989; Sussman, 1975; VanLehn, 1991a). There are several kinds of failures that may be involved, for example *expectation failures*, *retrieval failures*, or *knowledge application failures*. Expectation failures arise when the achieved outcome conflicts with expectations, regardless of whether the outcome was desirable (e.g., Collins & Birnbaum, 1988; Freed & Collins, 1993; Hammond, 1989; Leake, 1992; Owens, 1991; Ram & Cox, 1994/chapter 7; Schank, 1986). In our framework, an unexpected success is also treated as an expectation "failure." An example of a retrieval failure is the failure of a schema-based understanding program to retrieve an applicable schema, even though that schema exists in memory (Ram, 1993).

Knowledge application failures arise when retrieved knowledge structures fail to apply fully to new situations, and trigger learning to reconcile the conflicts (e.g., Kass & Leake, 1988; Leake, 1992; Mooney & Ourston, 1993; Park &

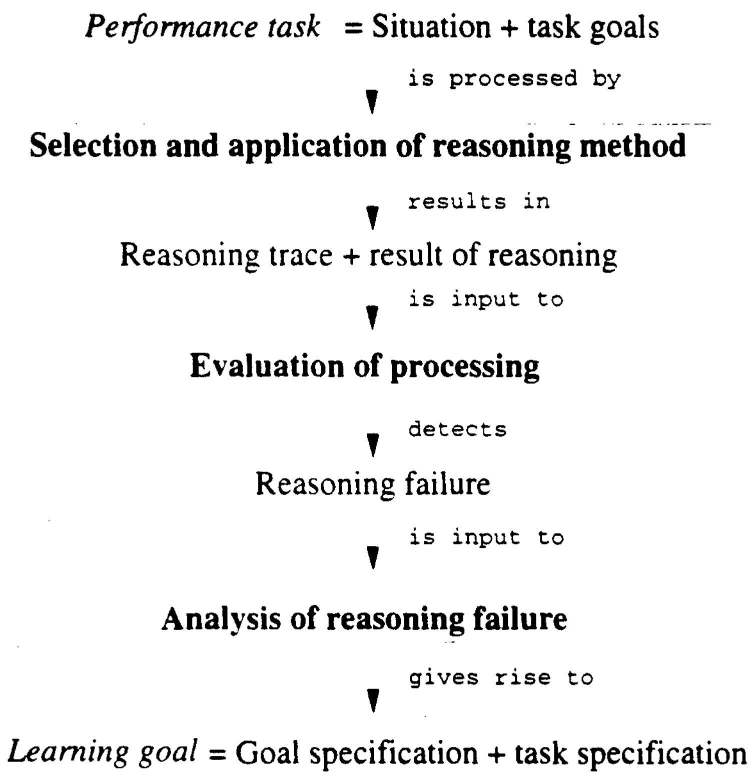

Performance task = Situation + task goals
is processed by
Selection and application of reasoning method
results in
Reasoning trace + result of reasoning
is input to
Evaluation of processing
detects
Reasoning failure
is input to
Analysis of reasoning failure
gives rise to
Learning goal = Goal specification + task specification

Figure 1 Generation of learning goals

Wilkins, 1990; Ram, 1993; Schank & Leake 1989). Ram, Cox, and Narayanan (chapter 18) present a taxonomy of possible types of failures and discuss their relationship to goal-driven learning.

Even if no failure has yet occurred, anticipation of a reasoning failure may trigger learning. For example, a reasoner may realize that it cannot perform a task and decide to perform the necessary learning before even attempting the task. In our framework, all these motivations for learning—reasoning failures, difficulties, impasses, suboptimalities, surprises, and other types of processing problems or anticipated processing problems—will be collectively and simply referred to as *failures*.

Different kinds of failures give rise to different kinds of learning goals. For example, a reasoner may need to acquire additional knowledge if its reasoning reached an impasse due to missing knowledge, as in the case of a novice stereo buyer who has no knowledge of which brand of stereo to buy. If the reasoner possessed sufficient knowledge but did not retrieve it at an appropriate time, it may need to reorganize memory (Ram & Cox, 1994/chapter 7; Ram, Cox, & Narayanan/chapter 18). A reasoner may need to modify the underlying representational vocabulary if its vocabulary is found to be inadequate (e.g., Schlimmer, 1987; Wrobel, 1988). In some situations, a reasoner might also need to add to its repertoire of reasoning strategies (e.g., Leake, 1993).

When a reasoning failure is detected, the reasoning trace is analyzed, in a process called credit/blame assignment, to find the source of the failure (Birnbaum, Collins, Freed & Krulwich, 1990; Hammond, 1989; Minsky, 1963; Ram & Cox, 1994/chapter 7; Weintraub, 1991). Blame assignment may be thought of as a process of model-based diagnosis of the reasoner itself (Birnbaum, Collins, Freed & Krulwich, 1990; Stroulia, Shankar, Goel, & Penberthy, 1992). If the failure is attributed to faulty knowledge, learning is needed to improve the reasoner's performance, and a learning goal is generated to repair that knowledge. In our framework, the learning goal is characterized in terms of two pieces of information: The desired learning—*what* learning is needed—and a description of the task that motivates learning—*why* learning is needed. The additional information about why learning is needed is important to allow the reasoner to carry out its tasks in an opportunistic manner, with learning goals (and the tasks that they support) being suspended until circumstances are favorable to their pursuit (Ram, 1989, 1991, 1993; Ram & Hunter, 1992/chapter 4).

Step 2: Pursuing learning goals: In the goal-driven view of learning, learning goals are treated analogously to task goals in the world. Just as task goals are achieved through a planning process using available methods for reasoning and action, learning goals are achieved through a *knowledge planning* process using available learning methods or strategies (Hunter, 1990/chapter 2; Quilici, in press; Ram & Hunter, 1992/chapter 4; Redmond, 1992). In the knowledge planning process, explicit reasoning is done about learning goals, their relative priorities, and strategies by which they can be achieved. These learning goals, also called *knowledge goals* (Ram, 1987, 1990; Ram & Hunter, 1992/chapter 4), can be represented in a goal dependency network (Michalski, 1993; Michalski & Ram/chapter 21), which is used to select and combine learning actions into learning strategies that are appropriate for current learning goals and for the learning opportunities provided by the current environment.

Individual learning actions may include performing knowledge acquisition (e.g., asking a friend to recommend a stereo) knowledge reorganization (e.g., grouping stores by the size of their stereo departments), knowledge reformulation or transmutation (e.g., forming new generalizations from stored episodes concerning others' experiences with particular sound equipment), and so on. Their application is guided by the learning goals of the reasoner (Gratch, DeJong, & Chien, 1994; Hunter, 1990/chapter 2; Michalski & Ram/chapter 21; Pryor & Collins, 1992/chapter 10; Ram & Cox, 1994/chapter 7; Ram & Hunter, 1992/chapter 4). Figure 2 sketches the second step of the goal-driven learning process. This step begins with reasoning about the relationships and relative priorities of learning goals in order to form a goal dependency network. Based on the information contained in the goal dependency network and on environmental factors affecting the appropriateness of different goals, the reasoner selects the learning goals to pursue. Learning strategies for achieving those goals in the current environment are then selected and applied.

Perspective on the framework: The model of learning embodied in the above steps contrasts with the approach to learning taken in traditional machine learning systems in artificial intelligence. Typically, in those systems, learning is primarily a passive, data-driven process of applying a single learning algorithm (or a predetermined combination of a few learning algorithms) to training examples presented to the system. Goal-driven learning, in contrast, is an active and strategic process driven by reasoning about information needs, alternative learning strategies, and opportunities in the environment. In our framework, the process of determining what to learn is an integral part of the computational model of learning, as is the process of deciding (on a dynamic basis) how and when to learn it.

Our view of goal-driven learning implies a tightly coupled relationship between learning and the "rest of reasoning." This view is consistent with recent models of intelligence that are framed as *integrated intelligent architectures* (sometimes known as *embedded systems*), in which the knowledge and reasoning tasks underlying learning and performance are integrated into a complete interacting system. Numerous approaches to such architectures were presented, for example, at the 1991 AAAI Spring Symposium on Integrated Intelligent Architectures (Laird, 1991), the 22nd Carnegie-Mellon Symposium on Cognition (VanLehn, 1991b) and at the Integrated Learning Workshop at the 1993 European Conference on Machine Learning (see Plaza, Aamodt, Ram, van de Velde,

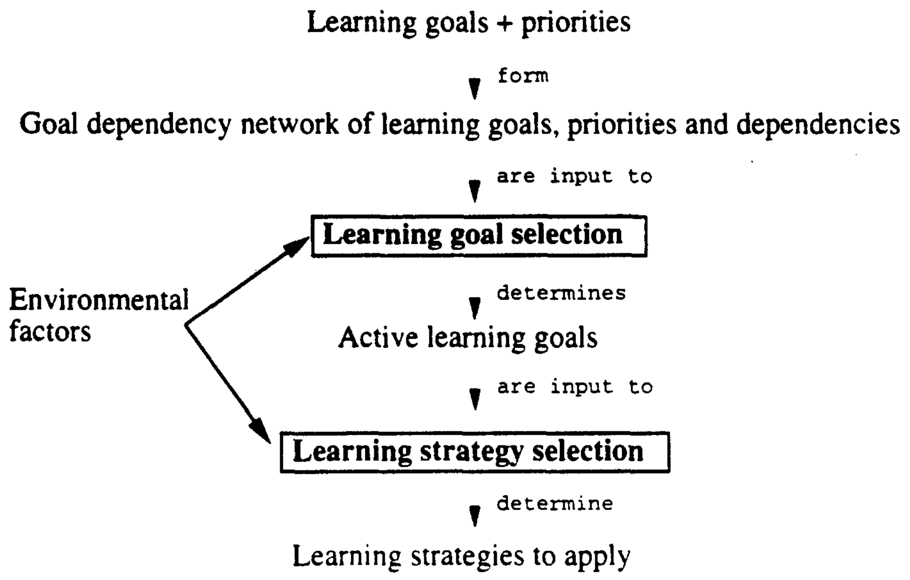

Learning goals + priorities
form
Goal dependency network of learning goals, priorities and dependencies
are input to
Learning goal selection
Environmental factors
determines
Active learning goals
are input to
Learning strategy selection
determine
Learning strategies to apply

Figure 2: Pursuing learning goals using appropriate learning strategies

& van Someren, 1993, for an overview). Some of these architectures propose that one or a few primitive mechanisms underly intelligence (e.g., Soar (Rosenbloom, Laird, & Newell, 1993), ACT* (Andersen, 1983)), while others integrate many (often higher-level) mechanisms that cooperate to achieve the agent's task and reasoning goals (e.g., PRODIGY (Carbonell, Etzioni, Gil, Joseph, Knoblock, Minton, & Veloso, 1991/chapter 11) and Theo (Mitchell, Allen, Chalasani, Cheng, Etzioni, Ringuette & Schlimmer, 1991)). A common theme in this research, and one that is compatible with the goal-driven learning framework proposed here, is the explicit representation of task goals, reasoning goals, and learning goals, and their role in a multistrategy reasoning process that integrates learning with performance tasks such as problem solving or comprehension (e.g., see Ram, Cox, & Narayanan/chapter 18). This theme is also shared with recent approaches that focus more on knowledge and knowledge-intensive reasoning than on the underlying cognitive architectures, such as Aamodt's (1991) CREEK and Hinrich and Kolodner's JULIA (Hinrichs, 1992).

5 Major issues in goal-driven learning

Our framework suggests several issues that must be addressed by a general theory of goal-driven learning:

- **What is a goal?**

  The term "goal" has been used to refer to several theoretical constructs in previous theories of learning and reasoning, including tasks, problem solving outcomes, desired states of the world, target concepts for learning, policies and orientations for learning, and so on. Consequently, characterizing goal-driven learning depends on determining the meaning of "goals" as they apply to the learning process.

- **What are the types of goals?**

  Given the wide range of goals and other influences described in learning research, another central issue is to identify different types of goals, how they relate to one another, and how different formulations of goal-based influences on learning can be placed in a unified framework.

- **How do goals influence processing and learning?**

  A premise of the goal-driven learning framework is that reasoning about goals directs the learning process. A fundamental question is what effects goals actually have on the learning process and how their influence is achieved.

- **What are the functional and pragmatic implications of goal-driven learning for the reasoner?**

  Given the differences between goal-driven learning and traditional learning models, one key question concerns the effects of goal-driven learning: What are the functional implications of goal-driven learning for the reasoner's own performance? What are the pragmatic implications of goal-driven learning as a model of reasoning, and for the design of intelligent systems?

- **What are the pragmatic implications of goal-driven learning as a cognitive model?**

  Considering goal-driven learning as a cognitive model raises questions about the pragmatic implications of that model. One key question concerns the predictions that the model suggests, which have implications for testing and validating theories of goal-driven learning. Another concerns the implications of the model for practical applications such as the design of instructional material and educational environments.

The following sections start with the first three of these points, illustrating relevant distinctions using examples from artificial intelligence research, and then take a broader view in considering the pragmatic ramifications of the goal-driven learning model. In Leake and Ram (1993/chapter 16), we return to these issues, discussing the individual perspectives on them that were advanced by the panelists at the Symposium on Goal-Driven Learning held at the 1992 Conference of the Cognitive Science Society.

6 What is a goal?

As Barsalou (chapter 17) observes, in some sense any reasoner executing a built-in procedure can be viewed as having a "goal" to perform that type of processing, so that any learner could be considered trivially "goal-driven". To distinguish between built-in behaviors and behaviors that are more explicitly goal-driven, Barsalou differentiates between implicit background orientations and explicit problem solving or task goals. Explicit task goals are the goals that guide a problem solving process in which a person intends to achieve a set of goals, assesses what must be performed to achieve them, and executes the needed actions. In contrast, an implicit background orientation is a behavior that is performed without explicit reasoning about when and how it should be pursued. For example, one such implicit orientation is the orientation to constantly maintain a world model that adequately represents the reasoner's environment (e.g., Barsalou/chapter 17, Leake, 1989, 1992), although in some formalisms this is expressed in terms of an explicit goal (see, e.g., Van de Velde, 1988).

Explicit goals are traditionally expressed as specifications of a target or desired outcome of a problem solving or learning task (e.g., Fikes, Hart, & Nilsson, 1972; Newell & Simon, 1972). However, Ram and Hunter (1992/chapter 4; Hunter, 1990/chapter 2; Ram & Cox, 1994/chapter 7) argue that capturing the introspective nature of the goal-driven learning process requires a richer characterization in which a goal is not merely a specification of a target. They argue that a target specification or an orientation is a goal only if the reasoner can actively plan to accomplish the goal, can make decisions about it, and can even decide to suspend it or not to pursue it.

In order for the reasoner to make such decisions, goals must be explicitly represented, and the reasoner must be able to reflect on its goals, how to achieve them, and their relative priorities and interdependencies. Ram and Hunter (1992/chapter 4) discuss a representation of learning goals in terms of the desired knowledge to be learned as well as the reason that the knowledge is needed. Additional representational issues concern the kinds of decision-making relationships that goals can enter into (Thagard & Millgram/chapter 19) and the intergoal relationships and interdependencies in which goals can play a role (Cox & Ram, 1994; Michalski & Ram/chapter 21; Schank and Abelson, 1977; Slade, 1993; Wilensky 1983).

7 Types of goals

In order to understand how goals can relate to one another and to learning, it is useful to consider the classes of goals that influence learning processes in existing computational models of learning (implemented as computer systems that learn). This section examines those classes. Our taxonomy is divided into the classes centered around *task goals*, *learning goals*, and *specifications, policies, and constraints*. Broadly, task goals determine *why* the reasoner is learning in the first place, learning goals specify *what* the reasoner needs to learn, and specification, policies and constraints influence *how* learning occurs.[1]

7.1 Task goals

In many systems, goals are modelled as descriptions of desired results or states in an external world, which we call "task goals." Task goals, exemplified by early planning programs such as STRIPS (Fikes, Hart, & Nilsson, 1972) and NOAH (Sacerdoti, 1977), are specifications of desired outcomes from a performance task in the external world, which are explicitly pursued through planful reasoning processes or, in some recent models, goal-directed reactive processes (e.g., Earl & Firby, 1994; Firby, 1987; Freed & Collins, 1994; Maes, 1990).

In order to pursue its task goals, a system may need to reason about the task goals and reasoning goals of other agents. In order to understand a story (or a real-world situation) involving other intelligent agents, a computer understanding system needs to model the goals and plans of those agents (Schank & Abelson, 1977; Wilensky, 1978); in addition, consideration of such goals affects the comprehension process of humans (Abbott & Black, 1986). Reasoning about other actors' goals also plays an important role in AI models of subjective understanding (Carbonell, 1983; Ram, 1990). Representation of goals and goal interactions is central in understanding as well as in planning (Wilensky, 1983). Such representations must capture both task goals and reasoning goals of both other agents and the system itself.

Because task goals characterize a desired state of affairs, they can also be used to describe the need for information that a planner requires to achieve that state of affairs (e.g., Etzioni, Hanks, Weld, Draper, Lesh, & Williamson, 1992; Leake, 1991b/chapter 9; Ram & Leake, 1991), to understand interactions between task goals (e.g., Freed & Collins, 1994), and to influence or bias learning strategies (e.g., Martin, 1994; Provost, 1994). In some models, task goals (and resulting learning goals) are decomposed into subgoals or task structures to facilitate planning and learning (e.g., Karlsson, 1994; Stroulia & Goel, 1994). In planning systems that store prior plans, such as Hammond's (1989) CHEF, task goals drive the search for relevant plans in memory and trigger learning of new plans and new indices for plan retrieval when failures arise. A similar role is played by "problem goals" in Veloso and Carbonell's (1993) model of case-based reasoning in the PRODIGY system, and by functional specifications or "design goals" in design programs (e.g., JULIA, Hinrichs, 1992; Kolodner, 1987).

[1]To clarify the differences and commonalities in different approaches, in this discussion we will use a common vocabulary and framework to discuss individual pieces of research in an attempt to present an integrated view of goal-driven learning, even at the expense of sometimes differing from the terminology used by the original researchers.

7.2 Learning goals

Other computational models explicitly describe goals for learning, rather than implicitly characterizing it in terms of the external task. These learning goals differ from task goals in that, while they too specify a desired state, the specified state is an internal or mental state—a state of knowledge or belief that the learner is attempting to achieve. Task goals are satisfied through problem solving in the external (usually physical) world, while learning goals are satisfied through a learning process that, in the goal-driven learning framework, is viewed as problem solving in the "informational" world. These learning goals have been characterized in the following ways:

- **Knowledge acquisition goals or knowledge goals:** Schank and Abelson (1977) describe a category of knowledge goals (called "D-KNOW" goals) to determine needed information. In their model, such goals arise when a planner requires knowledge of particular facts (e.g., the location of a desired object) to achieve its other goals. The planner generates plans for satisfying these goals using standard methods for seeking information in the external world.

  The term "knowledge goal" was introduced by Ram (1987), and general knowledge goals are discussed in more detail in Ram & Hunter (1992/chapter 4). Ram (1990) proposes the use of knowledge goals as the basis for focus of attention in understanding and learning. Hunter's (1990/chapter 2) IVY and INVESTIGATOR programs identify and pursue "knowledge acquisition goals" whose satisfaction constitutes learning in those systems. Ram and Cox's (1994/chapter 7) Meta-AQUA system uses knowledge goals such as knowledge refinement goals, knowledge reconciliation goals, and knowledge differentiation goals, to specify desired learning in a multistrategy learning system. The system then reasons about and selects the learning algorithms most appropriate for achieving its knowledge goals (Cox & Ram, 1994). Knowledge acquisition goals in desJardins's PAGODA system (called "learning goals" in that system) represent concepts which, if learned, would maximize the system's expected utility (DesJardins, 1992/chapter 8).

- **Questions:** Ram's (1991, 1993) AQUA asks "questions" which are then represented as "knowledge goals." As in IVY, AQUA's learning occurs through satisfying knowledge goals, but using different methods: IVY looks for desired information in diagnostic cases, and AQUA tries to answer its questions by reading stories (Ram & Hunter, 1992/chapter 4). In Oehlmann, Sleeman, and Edwards' IULIAN (1992), questions and experimentation interact in an exploratory discovery process applied to the domain of electrical circuits.

- **Learning goals:** Michalski's (1993) MTL framework uses "learning goals" as the starting point for learning; relationships between "learning goals" are then used to combine basic knowledge transmutations into learning actions and to prioritize learning activities. Such learning goals subsume knowledge acquisition goals, knowledge organization goals to reorganize existing knowledge, as in AQUA, Meta-AQUA, and IVY, and knowledge reformulation goals as in Meta-AQUA and MTL.

In our further discussion we will consider that learning goals, in addition to specifying the desired outcome of learning, specify the reason that the desired learning is required (for example, AQUA's "task specifications" specify the suspended reasoning task that is awaiting the knowledge to be learned).

7.3 Specifications, policies, and constraints

Numerous computer systems reflect other types of influences and constraints on learning that are goal-related. Although these are not properly "goals" in our sense, because they do not drive the learning process in an explicit manner, they may play an important role in influencing that process. One such influence is:

- **Goal concepts or target concepts:** Mitchell, Keller, and Kedar-Cabelli's (1986) EBG algorithm, implemented in several computer programs, takes as input a "target concept" or a "goal concept," in order to learn an operational description of that concept (see also Minton, 1990/chapter 3).

  "Target concepts" are similar to "learning goals" in that they specify the desired outcome of learning. However, in accordance with our earlier discussion of goals, target concepts are better viewed as specifications of the desired output of a learning strategy rather than an explicit goal to learn, unless they are pursued through active, strategic, or planful reasoning processes. In addition, target concepts (unlike learning goals) do not specify the motivation for learning.

The following categories are all ways to characterize the value of learning for the learner. In particular, they describe the policies under which the learning task should operate in order to better achieve the overarching learning goals, and describe relevant constraints on the processes that carry out the learning task:

- **Purposes:** Kedar-Cabelli's (1987) PURFORM uses "purposes" of artifacts, defined in terms of their role in enabling plans, to determine target concepts for learning. Leake's (1991b/chapter 9, 1992) ACCEPTER guides explanation evaluation in terms of tasks in the world which give rise to "purposes" to build particular types of explanations, which in turn provide the information needed to satisfy system learning goals.

- **Operationality criteria:** In explanation-based learning systems, "operationality criteria" (surveyed in Keller, 1988) characterize requirements for useful concept descriptions.
- **Preference criteria, inductive biases, and general policies and constraints:** Inductive learning systems such as PREDICTOR (Gordon & Perlis, 1989/chapter 13), LEX (Mitchell, 1982), and STABB (Utgoff, 1986) use an "inductive bias" to restrict the space of candidate hypotheses. Michalski's (1983) INDUCE method uses a lexicographic preference criterion to rank candidate hypotheses for generalization. Laird, Rosenbloom, and Newell's (1986) SOAR system incorporates a "policy" to learn from each subgoal during problem solving.
- **Utility metrics:** Minton's (1990/chapter 3) PRODIGY system uses a "utility metric" to determine whether a piece of knowledge is worth learning or storing. Gratch and DeJong's (1993) COMPOSER uses "expected utility" to characterize the quality of a reasoner with respect to a task, which increases with learning. desJardins's (1992/chapter 8) PAGODA computes the utility of plans and the costs of planning and learning to guide learning. PAGODA's "learning goals" represent concepts which, if learned, would maximize the system's expected utility.

Policies and constraints are not learning goals in the sense that the learner does not actively seek to satisfy them; instead, they influence the learning processes that the learner uses to achieve its learning goals. In particular, they describe the policies under which the learning task should operate in order to better achieve the overarching learning goals, and describe relevant constraints on the processes that carry out the learning task. Note that a learner might formulate explicit learning goals to learn these criteria. For example, a learner might formulate an explicit goal to learn appropriate biases for a given type of learning situation, and pursue an explicit learning agenda to learn such biases.

7.4 A unifying view

The underlying commonality among these constructs is that each reflects an intention to influence learning according to needs that are external to the learning process itself. However, quite different focuses are apparent in the formulations described in the previous sections. Consequently, developing a general theory of goal-driven learning depends on analyzing the relationships of these constructs and their role in reasoning and learning.

To relate the previous perspectives, we will establish a uniform vocabulary. In the following discussion, we will refer to the general class of *goals* to describe theoretical constructs that refer to mental entities reflecting desired states that are explicitly represented and actively pursued through a planful reasoning process.[2] *Task goals* will refer to goals which specify desired effects in the world external to the reasoner. *Learning goals* or *knowledge goals* will refer to goals which specify desired effects within the reasoner such as acquiring new knowledge or augmenting, reorganizing, or reformulating existing knowledge. Learning goals describe not only the desired processing outcome, but how the desired knowledge will be used when it is acquired. *Reasoning goals* will refer to more general internal goals to form conclusions or inferences through learning or other reasoning processes.

We will also refer to *target concepts* that specify a desired concept to be learned, but not necessarily learned through a goal-driven learning process: and to general *policies* or *orientations* that influence learning without being explicitly represented or available for manipulation by the reasoner's reasoning or learning process, including *constraints* on the formulation of hypotheses such as biases, operationality criteria, and utility metrics. Our vocabulary makes the following distinctions:

- **Task goals vs. reasoning goals:** Task goals are goals to be achieved in the world external to the reasoning system: reasoning goals are achieved within the reasoning mechanism of the system.

- **Reasoning goals vs. learning goals/knowledge goals:** Reasoning goals span the broad range of deliberative activities, including activities such as retrieval of relevant information, similarity assessment, etc. Learning goals/knowledge goals refer solely to goals to acquire or formulate particular types of knowledge.

- **Goals vs. policies/orientations/constraints:** Goals involve specifications of internal or external states to be actively planned for; policies specify background orientations that are implicit in the processes that achieve these states.

- **Learning goals vs. target concepts:** Learning goals are manipulated by an explicit, strategic planning process, while target concepts are specifications of desired results from a learning algorithm that uses the specification only to evaluate its results, rather than to guide the on-going learning process.

Table 3 summarizes these distinctions. Note, however, that these classes of goals can overlap and influence each other. Task goals have been used to guide learning and performance in several systems, and can also be used to formulate learning goals to acquire information necessary for a given task (Ram & Leake, 1991) or to come to a better understanding of the task itself (Freed & Collins,

[2]Note that this definition does not imply that goals or goal-driven processing must necessarily be conscious, nor that the reasoner must necessarily be able to report externally about this processing.

	Explicitly represented?	Range of effects (internal to reasoner or in external world)	Influences selection of solution algorithm?	Solution process	Effect on solution generation
Goals	Yes	Either	Yes	Planning	Guidance
Task goals	Yes	External	Yes	Planning actions in external world	Guidance
Reasoning goals/ learning goals	Yes	Internal	Yes	Knowledge planning	Guidance
Policies	Sometimes	Internal	Sometimes	Unspecified	Constraint
Target concepts	Yes	Internal	No	Unspecified	Guidance
Operationality criteria	Yes	Internal	No	Unspecified	Constraint

Table 3: Types of goals and policies.

1994). In conjunction with knowledge or theories, they can guide learning processes (Barsalou, 1991/chapter 5; Ng & Bereiter, 1991/chapter 14; Wisniewski & Medin, 1991/chapter 6). Likewise, although target concepts are generally provided to a learning system as input by a human user, in some models target concepts are generated from aspects of the performance task in a manner similar to the generation of learning goals. For example, Kedar-Cabelli (1987) discusses a method for generating target concepts from standard constraints on artifacts to be used in particular plans. Keller (1987) also sketches a process for generating learning goals from higher-level performance objectives. Similarly, policies (such as bias, which is usually formulated as a passive, background constraint on learning) may be actively monitored and modified by the reasoner to guide the learning task (Gordon & Perlis, 1989/chapter 13; Martin, 1994; Provost, 1994; Provost & Buchanan, 1992; Utgoff, 1986).

Several models include learning goals as an explicit part of their formulation of the learning process. Learning goals have been used to guide resource allocation, information search, hypothesis evaluation, and other aspects of learning; to select and combine learning strategies; to guide and to learn about the reasoning process itself; and to model active learning in educational contexts. These models are discussed in the following sections.

8 Role of goals in learning

Given the range of goals that can influence learning, it is not surprising that different models reflect different types and degrees of goal-based influence. Thagard and Millgram (chapter 19) propose a broad distinction between learning which is explicitly *goal-driven* and that which is *goal-relevant*. Goal-relevant processing is not explicitly directed by the goals of the reasoner, but results in outcomes that are nevertheless useful with respect to those goals. Thus goal-relevant processing is similar to Barsalou's (chapter 17) orientations in which the desired learning may occur as a side-effect of normal task-related processing. For example, a reasoner may have an implicit orientation to maintain an accurate model of the world around it (Barsalou, chapter 17; Leake, 1992). Goal-driven learning, in contrast, is driven by explicit learning goals of the reasoner: those goals influence or even determine the content of what is learned. As the reasoner's goals change, so does the learning that results.

One of the issues involved in goal-driven learning is how to balance competing goals to determine the goals and goal priorities that form the background for the goal-driven learning process (Thagard & Millgram, chapter 19). Once learning goals have been identified and prioritized, they can influence the system performance task, the system learning task, and the storage of results. The ways that goals can exert their influences are summarized in table 1, and are discussed in more detail in the following sections.

8.1 Guiding the performance task

In any goal-driven system, the influence of goals on the performance task also influences what is learned, by determining the focus of processing or changing the context in which learning is performed (Barsalou, chapter 17). For example, in case-based reasoning systems, the goals that drive processing also influence what is eventually learned (e.g., Kolodner, 1993; Hammond, 1989; Hinrichs, 1992; Ram, 1993; Veloso & Carbonell, 1993).

In addition, just as performance tasks can give rise to learning goals (e.g., the stereo buyer's learning goals), learning goals can themselves prompt and guide new performance tasks in service of the learning goals. For example, learning goals may guide tasks to gather needed information in the world, or to produce a situation in the world that is favorable to learning. Performance tasks may include reasoning tasks that are largely internal to the reasoner. For example, Leake (chapter 20) presents a model of explanation construction that is an integral part of a goal-driven reasoning and learning system. The reasoner can decide when explanations are needed, can characterize its information needs (goals), and can use this characterization to focus the search for explanations. Thus goals are used to guide the control procedure used in the performance task (explanation construction) and to manage the resources available for that task. Since explanation is a central part of learning, a goal-driven explainer is necessarily a goal-driven learner as well. Many performance tasks involve interaction with the outside world. For example, Ram (1991; Ram & Hunter, 1992/chapter 4) presents a model of natural language story understanding in which goal-driven processes are used to analyze and interpret natural language text (see also Carpenter & Alterman, 1994). Similarly, in Pryor and Collins's (1992/chapter 10) model, goals are used to guide the perception of visual images.

Xia and Yeung (1988/chapter 12) use goal-based considerations to learn new classifications of problem-solving strategies, and Hunter (1990/chapter 2) uses knowledge goals to guide the search for information by, for example, formulating appropriate queries to a database.

8.2 Guiding the learning task

The central tenet of goal-driven learning, and the thesis of this book, is that the learning is guided by explicit consideration of the reasoner's goals. The goal-driven learning framework involves first formulating explicit goals to find or infer certain beliefs, and then using these goals to drive reasoning and learning—which amounts to explicit decision-making and control. The resulting control of learning can be realized in several ways as described below.

8.2.1 Specifying the target of learning

Barsalou (1991/chapter 5) shows that people often derive categories in a dynamic, ad hoc manner during the construction of plans to achieve goals. For example, while foods are normally categorized into grains, vegetables, fruits, and so on, different category structures may be appropriate in the context of particular goals, giving rise to categories such as "foods to eat while on a diet". Some of these goal-derived categories become reasonably well-established for people or cultures in which the goal occurs often (such as dieting), but others remain ad hoc (such as "activities to do on a vacation in Japan with one's grandmother"). Thus learning can involve the construction of concepts that must be determined in a dynamic manner based on the demands of the particular task at hand. This is consistent with Ng and Bereiter's (1991/chapter 14) results on task- and goal-driven learning in an educational setting.

Wisniewski and Medin (1991/chapter 6) show that prior knowledge and intuitive theories can also influence learning (see also Murphy & Medin, 1985). They argue that tightly coupled interactions exist between knowledge and experience during learning. To the extent that learning is incremental, candidate hypotheses and theories learned earlier can influence later learning. In addition to previously learned theories, a goal-driven learner will also have previously formulated and possibly only partially satisfied learning goals as part of its learning context; these goals can also influence future learning (e.g., Ram, 1991, 1993).

In early artificial intelligence models of learning, goals were pre-specified as targets of particular learning algorithms. Such models did not have explicit learning goals; rather, they could be viewed as possessing background orientations to ensure that the learning actions are goal-relevant. However, task goals can be used to determine learning goals which specify the desired outcome of the learning task, whether it be a new piece of knowledge to be acquired or a new organization or formulation of existing knowledge (Ram & Leake, 1991). For example, Kedar-Cabelli (1987) and Keller (1987) propose extensions to earlier models in which the reasoner proposes its own targets rather than relying on an outside user to specify them.

In general, multiple learning goals are possible in any complex situation, and complex reasoning processes may be needed to determine which learning goals to generate (Krulwich, 1994; Ram, Cox, & Narayanan/chapter 18). It has been proposed that if learning is integrated with the reasoning process that it is in support of, an analysis of the reasoning process can be used to formulate learning goals (Hunter, 1990/chapter 2; Leake/chapter 20; Ram & Cox, 1994/chapter 7; Ram, Cox, & Narayanan/chapter 18; Ram & Hunter, 1992/chapter 4). For example, if a reasoner is unable to perform its task due to a missing piece of knowledge, it can formulate an explicit goal to learn that piece of knowledge. Learning goals may also seek to augment knowledge in other ways (e.g., learning a new antecedent to a rule (Mooney & Ourston, 1993; Park & Wilkins, 1990)), reorganize knowledge (e.g., learn a new index to an existing knowledge structure (Hammond, 1989; Ram, 1993)) or to reformulate existing knowledge (e.g., operationalization of abstract knowledge into a more directly usable form (Keller, 1988; Mostow, 1983); generalization or abstraction of examples (Michalski, 1993); modification of representational framework or vocabulary (Schlimmer, 1987; Wrobel, 1988)). Michalski (1993; Michalski & Ram/chapter 21) presents a taxonomy of the kinds of "knowledge transmutations" that may be used for various kinds of learning. As those learning goals are pursued, new learning goals may be generated on the fly, or existing learning goals abandoned, in response to changes in circumstances, the learner's knowledge, and overarching goals of the learner (Leake/chapter 20). Due to the dynamic nature of learning goals, the learning process itself must be dynamic as well.

8.2.2 Specifying the learning algorithms used

Many recent models of learning in humans and machines appeal to multiple methods for learning and reasoning in general (e.g., Michalski & Tecuci, 1993). For example, Ram, Narayanan, and Cox (1993) present a computational model of troubleshooting, based on a study of human troubleshooters on an electronics assembly line. In that model, several different learning strategies are used to improve troubleshooting performance, including compilation of knowledge through experience with specific problems, interactive acquisition of knowledge from a human expert, postponement of a problem until a later time, and forgetting knowledge that is made obsolete through changes in the manufacturing process.

In a wide range of artificial intelligence systems, goals determine the learning algorithms used to accomplish needed learning (e.g., Hunter, 1990/chapter 2; Michalski & Ram/chapter 21; Ram & Cox, 1994/chapter 7; Ram, Cox, & Narayanan, chapter 18; Ram & Hunter, 1992/chapter 4; Redmond, 1992; Quilici, in press; Xia & Yeung, 1988/chapter 12). A central issue in such models is how to select and combine appropriate learning algorithms in a given learning situation. This multistrategy learning process can be modeled with an explicit decision stage in which the appropriate learning strategy or strategies are identified (as shown in figure 2), followed by a strategy application stage in which the corresponding algorithm is executed (Ram & Cox, 1994/chapter 7). In addition, some models incorporate an explicit evaluatory phase in which the quality of a learned piece of knowledge is assessed. In those models, learning goals can be used to guide evaluation, such as in Leake's (1991b/chapter 9, chapter 20) use of goals to evaluate causal explanations.

8.2.3 Constraining the learning process

In some models of goal-driven learning, goals are used not to specify the desired target of learning or to select learning strategies, but rather to provide constraints to the process used for learning itself. For example, a reasoner that learns through inductive generalization must select from among a potentially very large number of possible hypotheses consistent with its inputs, and selecting the right candidate hypotheses can have an enormous effect on the ability of the reasoner to perform a particular task. Because the inputs do not adequately constrain the set of candidate hypotheses, hypothesis selection must be done via some criterion external to the inputs themselves. Such a criterion is called a "bias" (Mitchell, 1980) or "preference criterion" (Michalski, 1983). Although many early models of inductive learning appealed to a pre-determined bias, it is often advantageous for the learner to modify its bias (Utgoff, 1986). Gordon and Perlis (1989/chapter 13) discuss a computational model of inductive learning in which the formation of useful generalizations is facilitated by use of explicit biases; Provost and Buchanan (1992) discuss the use of "inductive policies" to automatically adjust the bias in a learning system based on explicitly represented tradeoffs; and Hadzikadic and Yun (1988) argue that concept formation should be viewed as a goal-driven, context-dependent process to assure its flexibility, efficiency, and generality.

Similarly, some computational models of analogical learning include a mechanism by which the reasoner's task goals can influence the analogical mapping process (e.g., Forbus & Oblinger, 1990; Kedar-Cabelli, 1987). As with other types of learning mechanisms, the functional justification for this influence is to ensure that the inferences made during learning are actually relevant to the overall performance task of the reasoner. Spellman and Holyoak (1993) present evidence for the influence of goals on analogical mapping in human learners as well.

8.2.4 Focusing search for information to carry out learning

Carrying out the desired learning may require acquisition of new information. Consequently, learning goals may give rise to processes that attempt to seek that information from the outside world. Some of the possible methods include reading text (Ram, 1991), querying a database (Hunter, 1990/chapter 2), active experimentation (Carbonell & Gil, 1990; Cohen, Kulikowski, & Berman, 1993; Rajamoney, 1993), or other planful activity to gather needed information (Etzioni, Hanks, Weld, Draper, Lesh, & Williamson, 1992; Leake/chapter 20; Pryor & Collins, chapter 10).

8.2.5 Determining when learning should be attempted

Having identified what to learn and how to learn it, a reasoner, in general, still needs to determine when (or whether) to perform the actions that will lead to the desired learning. This decision can be broken down into two fundamental aspects: identifying learning opportunities, and evaluating the potential utility of learning.

Identifying learning opportunities: It would be naive to expect the real world to be structured so as to facilitate the satisfaction of each individual's needs whenever they should arise. Instead, it is likely that goals (whether task goals or learning goals) will not be immediately satisfiable at the time when they are formulated. For example, a detective with the goal of identifying a criminal will usually need to do much investigation before having sufficient information to assign responsibility (Leake/chapter 20). Likewise, the reasoner may not have the resources to pursue all its goals all the time, forcing the reasoner to select particular goals to pursue (a detective performing multiple investigations will prioritize them according to their importance).

Furthermore, the real world environment may not provide the opportunity to pursue a particular goal even if the reasoner does decide to pursue it. For example, pursuit of a learning goal may require environmental resources (such as a library, or access to an expert) which may simply not be available at the time. In Ram's (1991) model of natural language story understanding, for example, the reasoner's questions about the story being read may not be answerable due to insufficient information being available in the story. Determining whether a suicide bombing is the work of a religious fanatic depends on having more information about the bomber than is likely to be available in the first accounts of the incident. Thus it is essential for the reasoner to be able to suspend its pursuit of a goal, and to be able to resume its learning processes when an opportunity to satisfy the goal presents itself (Ram & Hunter, 1992/chapter 4). An analogous argument has been made for the opportunistic pursuit of task or problem solving goals (Birnbaum & Collins, 1984; Hayes-Roth & Hayes-Roth, 1979; Hammond, Converse, Marks, & Seifert, 1993; Patalano, Seifert, & Hammond, 1993), and similar factors are relevant to learning behavior as well.

Evaluating the potential utility of learning: Not all learning is useful: learning may sometimes be undesirable if it leads to the accumulation of knowledge that is seldom used, that is not expressed in an efficient manner, or that impairs performance of the reasoner (Etzioni, 1992; Francis & Ram, 1993; Gratch & DeJong, 1993; Minton, 1990/chapter 3; Tambe, Newell, & Rosenbloom, 1990). Estimates of the potential utility of learned knowledge can be used to decide what to learn about, based on an analysis of the expected utility of the learning goals if they were to be achieved (e.g., desJardins, 1992/chapter 8). As mentioned earlier, while human learners may not be able to explicitly control their own reasoning processes to such a fine-grained level of detail, it is nevertheless possible to model human learning behavior in a goal-driven, utility-theoretic, or rational formalism (e.g., Anderson, 1991).

8.2.6 Evaluating the results of learning

The final criterion for the effectiveness of learning is how well the results of learning match the desired effects of the learning process. This question has received less attention than the questions of how goals influence initial learning. However, such evaluation has been used to determine whether to store the results of the learning algorithm (Minton, 1988; 1990/chapter 3), and could be used to formulate new learning goals in light of current results of the learning process.

8.3 Guiding storage

Evaluating the results of learning can a system to decide whether to store them, and, if it does, to decide how to store them. Minton (1988; 1990/chapter 3) proposes a learning process that estimates the usefulness of the generalized rules that it forms before they are actually stored in the rule library. The estimate is used to predict the utility of storing a new rule; only those rules expected to actually improve performance are stored. Once a rule has been stored, the effects of that rule on performance are monitored; rules that do not improve performance are removed from the rule library.

This kind of selective storage and retention of learned rules is an instance of a more general kind of goal-directed control of learning called *information filtering* (Markovitch & Scott, 1993). Information filters can be used to decide which learned items to store, which to retain in memory over time, which to apply in a given situation, and even which training experiences it should seek out and which it should learn from. The reasoner's task goals guide the filters in selecting what learning occurs and what products of learning are retained and used.

Despite the practical benefits of using goals to guide storage, claims of cognitive validity for such a model are controversial (see Barsalou/chapter 17 for arguments against direct goal-based guidance in human learners). However, regardless of whether explicit reasoning about goals affects individual storage decisions, goals can still have a profound effect on what results are stored by determining the course of processing. If goals determine processing and if processing determines storage, then goals determine storage indirectly. For example, in dynamic memory systems, processing intrinsically changes memory without the changes to memory necessarily being under explicit strategic control (Schank, 1982).

Goals also play a more direct role in storage in many AI systems. For example, case-based planning systems index learned plans according to the goals that those plans satisfy (e.g., Hammond, 1989; Hinrichs, 1992; Kolodner, 1987; Veloso & Carbonell, 1993). In this way, these systems organize their memories to facilitate re-use of those plans to accomplish similar goals. Likewise, case-based explanation systems index explanations according to the knowledge goals that the explanations were formulated to satisfy (e.g., Leake, 1991a, 1992; Ram, 1993; Schank & Leake, 1989). Thus, these systems attempt to store learned information to facilitate accomplishing similar future goals. Psychological experiments by Patalano, Seifert and Hammond (1993) also suggest that, in humans, goal-based factors can affect storage so as to facilitate noticing information relevant to the pursuit of those goals.

The preceding sections show that goal-driven learning provides flexibility for processing and the ability to tailor learning to current learner needs, helping to perform effective learning of useful information. Insofar as learning is simply a kind of reasoning (Ram, Cox, & Narayanan, chapter 18), many of the mechanisms of attention focussing and goal-driven processing in learning and in other reasoning will be identical. In particular, many of the results from research in planning may, mutatis mutandis, carry over to goal-driven learning as well.

9 Pragmatic implications of goal-driven learning

The previous discussion shows how goal-driven learning can provide considerable power in intelligent systems, whether those systems are viewed as computational models of human intelligence, or purely as artificial intelligence systems. In learning systems, goals can be used to focus learning and to avoid unrestricted search and inferencing. They can also be used to guide the information-seeking process and to make decisions about what, when, and how to learn.

Applying a planful model of learning promises to be fruitful for many applications, including perception (Pryor & Collins, 1992 10), intelligent information retrieval (Ram & Hunter, 1991), learning through apprenticeship (Redmond, 1992), knowledge acquisition (Quilici, in press), information search during explanation (Leake, chapter 20) medical diagnosis (Hunter, 1990), natural language understanding (Carpenter & Alterman, 1994; Cox & Ram, 1994; Ram, 1991), and manufacturing (Perez, 1994; Ram, Narayanan, & Cox, 1993).

In addition, goals can be used as a theoretical device to build computational models of strategic and active reasoning and learning processes, and such models have practical ramifications for the design of instructional material. Ng and Bereiter (1991/chapter 14) show that different kinds of goals facilitate different kinds of reasoning and result in different kinds of learning. Such results suggest principles for the design of computer-based tools for education (Scardamalia & Bereiter, 1991). For example, van Berkum, Hijne, de Jong, van Joolingen, and Njoo (1991/chapter 15) use goal-driven learning both as a theoretical framework for decomposing the education problem and as a guide toward designing simulation-based instructional software. Schank proposes that because of the importance of goals in motivating and guiding learning, instruction should be conducted using a particular type of simulation environment—a *goal-based scenario*—to exploit the role of learning goals (Schank, Fano, Jona, & Bell, 1993/1994). In goal-based scenarios, students play roles that are connected to their goals, and whose successful completion requires acquisition of the skills to be taught. In that way, goal-based scenarios provide a framework for students to perform goal-driven learning to acquire the skills to be taught.

To design educational environments that facilitate learning, one must understand the goal-driven information-seeking processes of the students who will be interacting with the environment, in order to encourage development of those processes. For example, in Scardamalia and Bereiter's (1991) Teacher C model, the teacher is concerned with helping students formulate their own goals, do their own activation of prior knowledge, ask their own questions, direct their own inquiry, and do their own monitoring of comprehension. Ng and Bereiter (1991/chapter 14) identify three types of goal orientation in learning: task completion, instructional, and knowledge-building. They show that students with knowledge-building goal orientation learn better—those students actively construct learning agendas, use prior knowledge in learning, and use the new learning to reconsider their prior knowledge.

Cognitive science research has shown that people learn by interpreting and constructing information (e.g., Resnick, 1983, 1987). Learning is viewed as a constructive, knowledge-building process (and often a collaborative one) rather than one of absorption (Bereiter, 1994; Roschelle, 1992). This principle has been used as the basis for the design of instructional scenarios which facilitate goal-driven interpretation and construction of knowledge (e.g., Edelson, 1993; Ng & Bereiter, 1991; Schank, Fano, Jona, & Bell, 1993/1994). Van Berkum, Hijne, de Jong, van Joolingen, and Njoo (1991/chapter 15) discuss learning environments in which computer simulations are used for instruction. They too distinguish between the learner's goals and the instructor's goals, and identify four aspects of the design of such systems: simulation models, learning goals, learning processes, and learning activity. In their model, learning goals have three dimensions: knowledge category (type of knowledge), knowledge representation (representation of that knowledge), and knowledge scope (generality and applicability of that knowledge). Learning occurs through interaction with simulated environments using four types of learning actions (orientation, hypothesis generation, testing, and evaluation) which are guided by the learning goals. As in Ng's model, learning is modelled as an active, constructive, and exploratory process, and the educational environment is designed to support such a process.

To this point, such applications of goal-driven learning models have been pursued independently of investigations of computer models of the goal-driven learning process itself. However, one of the goals of this volume is to show—by bringing together perspectives from diverse communities—the contributions that results from divergent research perspectives can make to one another, and to highlight the common questions that remain to be addressed. The previous discussion suggests the value of analyzing the goal-driven learning processes in humans as the basis for the design of instructional material and educational environments. In addition to its obvious practical value, this approach can be useful in empirically validating theories of goal-driven learning.

10 Summary

Computational and psychological investigations of goal-driven learning have addressed, broadly speaking, issues of what to learn, whether to learn, how to learn, and when to learn. In goal-driven learning, decisions about what, whether, how, and when to learn are determined by explicit reasoning about the reasoner's needs for information. Although many aspects of goal-driven learning have been investigated in diverse fields, that research has been conducted in a piecemeal fashion, largely segregated by field. Even when multiple studies have been conducted in a single field, as is the case for artificial intelligence, each study has tended to concentrate on a few aspects of the problem without placing those aspects within a unifying framework and examining their larger implications.

This chapter has presented a unifying picture of existing goal-driven learning research in terms of a new framework for modeling goal-driven learning, in terms of the types of goals that may guide learning, and in terms of the ways those goals can influence learning. The chapter has also discussed some of the pragmatic ramifications of the goal-driven learning model, both for intelligent systems and for educational applications. Those ramifications provide motivations for advancing our understanding of the goal-driven learning process.

The framework presented here is not suggested as a final theory of goal-driven learning, but rather a device for understanding the relationships of different results relevant to goal-driven learning and for suggesting issues that must be addressed with further investigation through a coordinated multidisciplinary research effort. The individual models and perspectives of the following chapters illuminate specific aspects of the framework and the issues that remain to be addressed in future research.

Acknowledgements

This research was supported in part by the National Science Foundation under grants IRI-9009710 and IRI-9409348, by the Air Force Office of Sponsored Research under contract F49620-94-1-0092, by the Georgia Institute of Technology, and by Indiana University. We would like to thank Betty Stanton for her enthusiasm and support for our work on this volume, and to thank the MIT Press anonymous reviewers for their detailed comments on this and the other chapters in this volume, as well as on the composition of the volume itself. We would also like to thank Michael Cox, Eric Domeshek, Susan Fox, Anthony Francis, Janet Kolodner, Kenny Moorman, Paul Panaro, and Raja Sooriamurthi for their many helpful comments on earlier drafts of this chapter.

References

Aamodt, A. (1991). *A Knowledge Intensive Approach to Problem-Solving and Sustained Learning* . Ph.D. Dissertation. University of Trondheim. Norwegian Institute of Technology, May, 1991.

Abbot, V., & Black, J. (1986). Goal-Related Inferences in Comprehension. In J. Galambos, R. Abelson, & J. Black, editors, *Knowledge Structures*, pages 123-142, Lawrence Erlbaum Associates, Hillsdale, NJ.

Anderson, J. (1983). *The Architecture of Cognition*, Harvard University Press, Cambridge, MA.

Anderson, J. (1991). The Place of Cognitive Architectures in a Rational Analysis. In K. VanLehn, editor, *Architectures for Cognition*, pages 1–24, Lawrence Erlbaum Associates, Hillsdale, NJ.

Barsalou, L. (1991/chapter 5 of this volume). Deriving Categories to Achieve Goals. In G.H. Bower, editor, *The Psychology of Learning and Motivation: Advances in Research and Theory*, Volume 27, Academic Press, New York, NY.

Barsalou (chapter 17 of this volume). Storage Side Effects: Studying Processing to Understand Learning. In A. Ram & D.B. Leake, editors, *Goal-Driven Learning*, MIT Press/Bradford Books, Cambridge, MA.

Bereiter, C. (1994). Paper presented at the AAAI Spring Symposium on Goal-Driven Learning, Stanford, CA.

Birnbaum, L. & Collins, G. (1984). Opportunistic Planning and Freudian Slips. In *Proceedings of the Sixth Annual Conference of the Cognitive Science Society*, pages 124–127, Boulder, CO.

Birnbaum, L., Collins, G., Freed, M., & Krulwich, B. (1990). Model-Based Diagnosis of Planning Failures. In *Proceedings of the Eighth National Conference on Artificial Intelligence*, pages 318–323, Boston, MA.

Carbonell, J.G. (1986). Derivational Analogy: A Theory of Reconstructive Problem Solving and Expertise Acquisition. In R.S. Michalski, J.G. Carbonell, & T.M. Mitchell, editors, *Machine Learning II: An Artificial Intelligence Approach*, pages 371–392, Morgan Kaufman Publishers, San Mateo, CA.

Carbonell, J.G., Etzioni, O., Gil, Y., Joseph, R., Knoblock, C., Minton, S., & Veloso, M. (1991/chapter 11). Planning and Learning in PRODIGY: Overview of an Integrated Architecture. *SIGART Bulletin*, 2(4).

Carbonell, J.G. & Gil, Y. (1990) Learning by Experimentation: The Operator Refinement Method. In Y. Kodratoff and R. Michalski, editors, *Machine Learning III: An Artificial Intelligence Approach*, pages 191–213, Morgan Kaufman Publishers, San Mateo, CA.

Carpenter, T. & Alterman, R. (1994). *A Reading Agent*. In *Proceedings of the Twelfth National Conference on Artificial Intelligence*, pages 63–67.

Chien, S. (1989). Using and Refining Simplifications: Explanation-based Learning of Plans in Intractable Domains. In *Proceedings of the Eleventh International Joint Conference on Artificial Intelligence*, pages 590-595, Detroit, MI.

Cohen. D., Kulikowski, C., & Berman. H. (1993). Knowledge-Based Generation of Machine Learning Experiments: Learning with DNA Crystallography Data. In *Proceedings of the Intelligent Systems for Molecular Biology Symposium*. pages 92–100.

Collins, G. & Birnbaum, L. (1988). An Explanation-Based Approach to the Transfer of Planning Knowledge Across Domains. In *Proceedings of the 1988 AAAI Spring Symposium on Explanation-based Learning*, pages 107–111, Stanford. CA.

Collins, G., Birnbaum. L., Krulwich, B. & Freed, M. (1991). Plan Debugging in an Intentional System. In *Proceedings of the Twelfth International Joint Conference on Artificial Intelligence*. pages 353-358. Sydney, Australia.

Cox. M.T. (1993). *Introspective Multistrategy Learning*. Technical Report GIT-CS-93/02. Cognitive Science Program. College of Computing. Georgia Institute of Technology. Atlanta. GA.

Cox. M.T. & Ram. A. (1994). Managing Learning Goals in Strategy Selection Problems. In *Proceedings of the Second European Workshop on Case-Based Reasoning*. pages 85–93. Chantilly. France.

Dennett. D. (1987). *The Intentional Stance*. Bradford Books/MIT Press. Boston. MA.

desJardins. M. (1992/chapter 8 of this volume). Goal-Directed Learning: A Decision-Theoretic Model for Deciding What to Learn Next. In *Proceedings of the ML-92 Workshop on Machine Discovery*. pages 147–151. Ninth International Machine Learning Conference. University of Aberdeen. Scotland.

Earl. C. & Firby. R.J. (1994). An Integrated Action and Learning System. In M. desJardins & A. Ram. editors. *Proceedings of the AAAI Spring Symposium on Goal-Driven Learning*. pages 22–27. Stanford. CA.

Edelson. D.C. (1993). *Learning from Stories: Indexing and Reminding in a Socratic Case-Based Teaching System for Elementary School Biology*. Ph.D. dissertation. Technical Report #43. Northwestern University. Institute of the Learning Sciences. Chicago. IL.

Etzioni. O. (1992). An Asymptotic Analysis of Speedup Learning. In *Machine Learning: Proceedings of the Ninth International Workshop*. pages 129–136. 1992.

Etzioni. O.. Hanks. S., Weld. D . Draper. D., Lesh. N., & Williamson. M. (1992). An Approach to Planning with Incomplete Information. In *Proceedings of the International Conference on Knowledge Representation*. Morgan Kaufmann. San Francisco. CA.

Faries, J. & Reiser. B. (1988). Access and Use of Previous Solutions in a Problem Solving Situation. In *Proceedings of the Tenth Annual Conference of the Cognitive Science Society*. pages 433–439. Montreal.

Firby. J. (1987). Adaptive Execution in Complex Dynamic Worlds. Ph.D. Dissertation. Research Report #672. Yale University. Department of Computer Science. New Haven. CT

Fikes, R.E., Hart, P.E., & Nilsson, N.J. (1972). Learning and Executing Generalized Robot Plans. *Artificial Intelligence*, 3:251–288..

Forbus, K.D. & Oblinger, D. (1990). Making SME Greedy and Pragmatic. In *Proceedings of the Twelfth Annual Conference of the Cognitive Science Society*, pages 61–68, Cambridge, MA.

Forrest-Pressley, D.L., MacKinnon, G.E., & Waller, T.G. (1985), editors. *Metacognition, Cognition, and Human Performance, Volume 1 (Theoretical Perspectives)*, Academic Press, New York.

Francis, A. & Ram, A. (1993). Computational Models of the Utility Problem and their Application to a Utility Analysis of Case-Based Reasoning. In *Proceedings of the ML-93 Workshop on Knowledge Compilation and Speedup Learning*, Tenth International Machine Learning Conference, University of Massachusetts, Amherst, MA.

Freed, M. & Collins, G. (1993). A Model-Based Approach to Learning from Attention-Focusing Failures. In *Proceedings of the Fifteenth Annual Conference of the Cognitive Science Society*, pages 434–439, Boulder, CO.

Freed, M. & Collins, G. (1994). Learning to Cope with Task Interactions. In M. desJardins & A. Ram, editors, *Proceedings of the AAAI Spring Symposium on Goal-Driven Learning*, pages 28–35, Stanford, CA.

Gauld, C. (1986). Models, meters and memory. *Research in Science Education*, 16:49–54.

Gavelek, J.R. & Raphael, R.E. (1985). Metacognition, Instruction and the Role of Questioning Activities. In D.L. Forrest-Pressley, G.E. MacKinnon, & T.G. Waller, editors, *Metacognition, Cognition, and Human Performance, Volume 2 (Instructional Practices)*, pages 103–136, Academic Press, New York.

Gordon, D. & Perlis, D. (1989/chapter 13). Explicitly Biased Generalization. *Computational Intelligence*, 5(2):67-81.

Graesser, A.C., Person, N., & Huber, J. (1992). Mechanisms that Generate Questions. In T.W. Lauer, E. Peacock, and A.C. Graesser, editors, *Questions and Information Systems*, pages 167–187, Lawrence Erlbaum Associates, Hillsdale, NJ.

Gratch, J. & DeJong, G. (1993). Assessing the Value of Information to Guide Learning Systems. In *Proceedings of the ML-93 Workshop on Knowledge Compilation and Speedup Learning*, pages 65–71, Tenth International Machine Learning Conference, University of Massachusetts, Amherst, MA.

Gratch, J., DeJong, G., & Chien, S.A. (1994). Deciding When and How to Learn. In M. desJardins & A. Ram, editors, *Proceedings of the AAAI Spring Symposium on Goal-Driven Learning*, pages 36–45, Stanford, CA.

Greeno, J.G. & Simon, H.A. (1988). Problem Solving and Reasoning. In R.C. Atkinson, H. Hernstein, G. Lindzey, & R.D. Luce, editors, *Stevens' Handbook of Experimental Psychology, Vol. 2: Learning and Cognition*, pages 589–673, Wiley, New York.

Hadzikadic, M. & Yun, D.Y.Y. (1988). Concept Formation by Goal-Driven.

Context-Dependent Classification. In Z.W. Ras & L. Saitta, editors, *Methodologies for Intelligent Systems*, 3:322–332, 1988.

Hammond, K.J. (1989). *Case-Based Planning: Viewing Planning as a Memory Task*. Academic Press, Boston, MA.

Hammond, K., Converse, T., Marks, M., & Seifert, C.M. (1993). Opportunism and Learning. *Machine Learning*, 10(3):279–309.

Hayes-Roth, B. & Hayes-Roth, F. (1979). A Cognitive Model of Planning. *Cognitive Science*, 3(4):275-310.

Hayes-Roth, F. & Lesser, V. (1976). Focus of attention in a distributed logic speech understanding system. In *Proceedings of the IEEE International Conference on Accoustics, Speech and Signal Processing*, pages 416–420. Philadelphia, PA.

Hinrichs, T. (1992). *Problem Solving in Open Worlds: A Case Study in Design*. Lawrence Erlbaum Associates, Hillsdale, NJ.

Hoffman, C., Mischel, W., & Mazze, K. (1981). The Role of Purpose in the Organization of Information about Behavior: Trait-based versus Goal-Based Categories in Human Cognition. *Journal of Personality and Social Psychology*, 39:211–255.

Hunter, L.E. (1990/chapter 2 of this volume). Planning to Learn. In *Proceedings of the Twelfth Annual Conference of the Cognitive Science Society*, pages 261–276, Boston, MA.

Karlsson, J. (1994). Task Decomposition in Reinforcement Learning. In M. desJardins & A. Ram, editors, *Proceedings of the AAAI Spring Symposium on Goal-Driven Learning*, pages 46–53, Stanford, CA.

Kass, A. & Leake, D.B. (1988). Case-Based Reasoning Applied to Constructing Explanations. In J.L. Kolodner, editor, *Proceedings of the Case-Based Reasoning Workshop*, pages 190–208, Morgan Kaufmann Publishers, San Mateo, CA.

Keller, R. (1987). The Role of Explicit Contextual Knowledge in Learning Concepts to Improve Performance. Ph.D. Dissertation, Technical Report ML-TR-7, Department of Computer Science, Stanford University.

Keller, R. (1988). Defining Operationality for Explanation-Based Learning. *Artificial Intelligence*, 35:227–241.

Kedar-Cabelli, S. (1987). Formulating Concepts According to Purpose. In *Proceedings of the Sixth Annual National Conference on Artificial Intelligence*, pages 477-481, Seattle, WA.

Kintsch, W. (1988). The Role of Knowledge in Discourse Comprehension: A Construction-Integration Model. *Psychological Review*, 95(2):163–182.

Kocabas, S. (1994). Goal-Directed Discovery and Explanation in Particle Physics. In M. desJardins & A. Ram, editors, *Proceedings of the AAAI Spring Symposium on Goal-Driven Learning*, pages 54–61, Stanford, CA.

Kolodner, J.L. (1987). Extending Problem-Solving Capabilities through Case-Based Inference. In *Proceedings of the Fourth International Workshop on Machine Learning*, Irvine, CA.

Kolodner, J.L. (1993). *Case-Based Reasoning*. Morgan Kaufmann Publishers, San Mateo, CA.

Krulwich, B. (1994). A Component-Model Approach to Determining What to Learn. In M. desJardins & A. Ram, editors, *Proceedings of the AAAI Spring Symposium on Goal-Driven Learning*, pages 62–71, Stanford, CA.

Krulwich, B., Birnbaum, L., & Collins, G. (1992). Learning Several Lessons from one Experience. In *Proceedings of the Fourteenth Annual Conference of the Cognitive Science Society*, pages 242–247, Bloomington, IN.

Laird, J.E. (1991), editor. Special Section on Integrated Cognitive Architectures. *SIGART Bulletin*, 2(4):12–184.

Laird, J.E., Rosenbloom, P.S., & Newell, A. (1986). Chunking in Soar: The Anatomy of a General Learning Mechanism. *Machine Learning*, 1:11–46.

Leake, D.B. (1989). Anomaly Detection Strategies for Schema-Based Story Understanding. In *Proceedings of the Eleventh Annual Conference of the Cognitive Science Society*, pages 490–497, Ann Arbor, MI.

Leake, D.B. (1991a). An Indexing Vocabulary for Case-Based Explanation. In *Proceedings of the Ninth National Conference on Artificial Intelligence*, Anaheim, CA, pages 10–15.

Leake, D.B. (1991b/chapter 9 of this volume). Goal-Based Explanation Evaluation. *Cognitive Science*, 15:509–545.

Leake, D.B. (1992). *Evaluating Explanations: A Content Theory*. Lawrence Erlbaum Associates, Hillsdale, NJ.

Leake, D.B. (1993). Learning Adaptation Strategies by Introspective Reasoning about Memory Search. In D.B. Leake, editor, *Proceedings of the AAAI-93 Workshop on Case-Based Reasoning*, pages 57–63, AAAI Press, Menlo Park, CA.

Leake, D.B. & Ram, A. (1993/chapter 16 of this volume). *Goal-Driven Learning: Fundamental Issues (A Symposium Report)*. *AI Magazine*, 14(4).

Leake, D.B. (chapter 20 of this volume). Goal-Driven Integration of Explanation and Action. In A. Ram & D.B. Leake, editors, *Goal-Driven Learning*, MIT Press/Bradford Books, Cambridge, MA.

Maes, P. (1990). Situated Agents Can Have Goals. *Robotics and Autonomous Systems*, 6:49–70.

Markovitch, S. & Scott, P.D. (1993). Information Filtering: Selection Mechanisms in Learning Systems. *Machine Learning*, 10:113–151.

Martin, J.D. (1994). DCI Supervised and Unsupervised Clustering. In M. desJardins & A. Ram, editors, *Proceedings of the AAAI Spring Symposium on Goal-Driven Learning*, pages 80–87, Stanford, CA.

Michalski, R.S. (1983). A Theory and Methodology of Inductive Learning. *Artificial Intelligence*, 20:111-161

Michalski, R.S. (1993). Inferential Theory of Learning as a Conceptual Basis for Multistrategy Learning. *Machine Learning*, 11(2/3):111–151, 1993.

Michalski, R.S. & Ram, A (chapter 21 of this volume). Learning as Goal-Driven Inference. In A. Ram & D.B. Leake, editors, *Goal-Driven Learning*, MIT Press/Bradford Books, Cambridge, MA.

Michalski, R.S. & Tecuci, G. (1994), editors. *Machine Learning: A Multistrategy Approach, Volume IV*, Morgan Kaufman Publishers, San Mateo, CA.

Minsky, M. (1963). Steps towards Artificial Intelligence. In E.A. Feigenbaum & J. Feldman, editors, *Computers and Thought*, pages 406–450, McGraw-Hill, New York, NY.

Minton, S. (1988). *Learning Search Control Knowledge: An Explanation-Based Approach*. Kluwer Academic Publishers, Boston.

Minton, S. (1990/chapter 3 of this volume). Quantitative Results Concerning the Utility of Explanation-Based Learning. *Artificial Intelligence*, 42:363-391.

Mitchell, T.M. (1980). *The Need for Biases in Learning Generalizations*. Technical Report CBM-TR-117, Rutgers University, New Brunswick, NJ.

Mitchell, T.M. (1982). Generalization as Search. *Artificial Intelligence*, 18:203–226.

Mitchell, T.M., Allen, J., Chalasani, P., Cheng, J., Etzioni, O., Ringuette, M. & Schlimmer, J. (1991). Theo: A Framework for Self-Improving Systems. In K. VanLehn, editor, *Architectures for Intelligence: The Twenty-Second Carnegie Mellon Symposium on Cognition*, pages 323–355, Lawrence Erlbaum Associates, Hillsdale, NJ.

Mitchell, T.M. & Keller, R. (1983). Goal Directed Learning. In *Proceedings of the International Machine Learning Workshop*, pages 117–118, Monticello, IL.

Mitchell, T.M., Keller, R., & Kedar-Cabelli, S. (1986). Explanation-Based Generalization: A Unifying View. *Machine Learning*, 1(1):47–80.

McKoon, G. & Ratcliff, R. (1992). Inference During Reading. *Psychological Review*, 99: 440-466.

Mooney, R. (1990). *A General Explanation-based Learning Mechanism and its Application to Narrative Understanding*. Morgan Kaufmann Publishers, San Mateo, CA.

Mooney, R.J. & Ourston, D. (1993). A Multistrategy Approach to Theory Refinement. In R.S. Michalski & G. Tecuci, editors, *Machine Learning: A Multistrategy Approach, Volume IV*, pages 141–164, Morgan Kaufman Publishers, San Mateo, CA.

Mostow, J. (1983). Machine Transformation of Advice into a Heuristic Search Procedure. In R.S. Michalski, J.G. Carbonell, and T.M. Mitchell, editors, *Machine Learning: An Artificial Intelligence Approach*, pages 367–403, Morgan Kaufmann Publishers, San Mateo, CA.

Mostow, J. & Bhatnagar, N. (1987). FAILSAFE—A Floor Planner that Uses EBG to Learn from its Failures. In *Proceedings of the Tenth International Joint Conference on Artificial Intelligence*, pages 249–255, Milan, Italy.

Murphy, G.L. & Medin, D.L. (1985). The Role of Theories in Conceptual Coherence. *Psychological Review*, 92:289–316.

Newell, A. (1990). *Unified Theories of Cognition*. Harvard University Press, Cambridge, MA.

Newell, A. & Simon, H. (1972). *Human Problem Solving.* Prentice-Hall, Englewood Cliffs, NJ.

Ng, E. & Bereiter, C. (1991/chapter 14 of this volume). Three Levels of Goal Orientation in Learning. *The Journal of the Learning Sciences*, 1(3&4):243–271.

Oehlmann, R., Sleeman, D., & Edwards, P. (1992.) Self-Questioning and Experimentation in an Exploratory Discovery System. In *Proceedings of the ML-92 Workshop on Machine Discovery*, pages 41–50, Ninth International Machine Learning Conference, University of Aberdeen, Scotland.

Owens, C. (1991). A Functional Taxonomy of Abstract Plan Failures. In *Proceedings of the Thirteenth Annual Conference of the Cognitive Science Society*, pages 167–172, Chicago, IL.

Patalano, A., Seifert, C., & Hammond, K. (1993). Predictive Encoding: Planning for Opportunities. *Proceedings of the Fifteenth Annual Conference of the Cognitive Science Society*, pages 800–805, Boulder, CO.

Park, Y. & Wilkins, D. (1990). Establishing the Coherence of an Explanation to Improve Refinement of an Incomplete Knowledge Base. In *Proceedings of the Eighth National Conference on Artificial Intelligence*, pages 511-516, Boston, MA.

Perez, M.A. (1994). The Goal is to Produce Better Plans. In M. desJardins & A. Ram, editors, *Proceedings of the AAAI Spring Symposium on Goal-Driven Learning*, pages 88–93, Stanford, CA.

Plaza, E., Aamodt, A., Ram, A., Van de Welde, W., & van Someren, M. (1993). Integrated Learning Architectures. In *Proceedings of the European Conference on Machine Learning*, Vienna, Austria.

Provost, F.J. (1994). Goal-Directed Inductive Learning: Trading off Accuracy for Reduced Error Cost. In M. desJardins & A. Ram, editors, *Proceedings of the AAAI Spring Symposium on Goal-Driven Learning*, pages 94–101, Stanford, CA.

Provost, F.J. & Buchanan, B.G. (1992). Inductive Policy. In *Proceedings of the Tenth National Conference on Artificial Intelligence*, pages 255–261, San Jose, CA.

Pryor, L. & Collins, G. (1992/chapter 10 of this volume). Planning to Perceive. In *Proceedings of the 1992 AAAI Spring Symposium on Selective Perception*, Stanford, CA.

Quilici, A. (in press). Toward Automatic Acquisition of an Advisory System's Knowledge Base. *Applied Intelligence*.

Rajamoney, S. (1993). Designing Experiments to Extend the Domain Theory. In DeJong, G., editor, *Investigating Explanation-Based Learning*, Kluwer, Boston.

Ram, A. (1987). AQUA: Asking Questions and Understanding Answers. In *Proceedings of the Sixth Annual National Conference on Artificial Intelligence*, pages 312–316, Seattle, WA

Ram, A. (1989). *Question-Driven Understanding: An Integrated Theory of Story Understanding, Memory and Learning.* Ph.D. Dissertation, Research Report

#710, Yale University, Department of Computer Science, New Haven, CT.

Ram, A. (1990). Knowledge Goals: A Theory of Interestingness. In *Proceedings of the Twelfth Annual Conference of the Cognitive Science Society*, pages 206–214, Cambridge, MA.

Ram, A. (1991). A Theory of Questions and Question Asking. *The Journal of the Learning Sciences*, 1(3&4):273–318.

Ram, A. (1993). Indexing, Elaboration and Refinement: Incremental Learning of Explanatory Cases. *Machine Learning*, 10:201–248.

Ram, A. & Cox, M.T. (1994/chapter 7 of this volume). Introspective Reasoning using Meta-Explanations for Multistrategy Learning. In R.S. Michalski & G. Tecuci, editors, *Machine Learning: A Multistrategy Approach, Volume IV*. Morgan Kaufman Publishers, San Mateo, CA.

Ram, A., Cox, M.T., & Narayanan, S. (chapter 18 of this volume). Goal-Driven Learning in Multistrategy Reasoning and Learning Systems. In A. Ram & D.B. Leake, editors, *Goal-Driven Learning*. MIT Press/Bradford Books, Cambridge, MA.

Ram, A. & Hunter, L. (1992/chapter 4 of this volume). The Use of Explicit Goals for Knowledge to Guide Inference and Learning. *Applied Intelligence*, 2(1):47–73.

Ram, A. & Leake, D.B. (1991). Evaluation of Explanatory Hypotheses. In *Proceedings of the Thirteenth Annual Conference of the Cognitive Science Society*, pages 867–871, Chicago, IL.

Ram, A., Narayanan, S., & Cox, M.T. (1993). Learning to troubleshoot: Multistrategy learning of diagnostic knowledge for a real-world problem solving task. Technical Report GIT-CC-93/67, College of Computing, Georgia Institute of Technology, Atlanta, Georgia 30332-0280, 1993.

Redmond, M. (1992). *Learning by Observing and Understanding Expert Problem Solving*. Ph.D. Dissertation, Technical report GIT-CC-92/43, College of Computing, Georgia Institute of Technology, Atlanta, GA.

Resnick, L.B. (1983). Towards a Cognitive Theory of Instruction. In S.G. Paris, G.M. Olson, & H.W. Stevenson, editors, *Learning and Motivating in the Classroom*, pages 6–38, Lawrence Erlbaum Associates, Hillsdale, NJ.

Resnick, L.B. (1987). Constructing Knowledge in School. In L.S. Liben, editor, *Development and Learning: Conflict or Congruence*, pages 19-50, Lawrence Erlbaum Associates, Hillsdale, NJ.

Riesbeck, C. (1981). Failure-driven Reminding for Incremental Learning. In *Proceedings of the Seventh International Joint Conference on Artificial Intelligence*, pages 115–120, Vancouver, British Columbia.

Roschelle, J. (1992). Learning by Collaborating: Convergent Conceptual Change. *The Journal of the Learning Sciences*, 2(3):235–276.

Rosenbloom, P., Laird, J., & Newell, A. (1993) *The Soar papers: Research on integrated intelligence*. MIT Press, Cambridge, MA.

Sacerdoti, E.D. (1975). *A Structure for Plans and Behavior*. Technical Note #109, SRI International. Summarized in P.R. Cohen & E.A. Feigenbaum, *Handbook of AI, Volume III*, pages 541–550.

Scardamalia, M. & Bereiter, C. (1991). Higher Levels of Agency for Children in Knowledge Building: A Challenge for the Design of New Knowledge Media. *The Journal of the Learning Sciences*, 1(1):37–86.

Schank, R.C. (1982). *Dynamic Memory: A Theory of Learning in Computers and People*. Cambridge University Press, Cambridge, England.

Schank, R.C. (1986). *Explanation Patterns: Understanding Mechanically and Creatively*. Lawrence Erlbaum Associates, Hillsdale, NJ.

Schank, R.C. & Abelson, R. (1977). *Scripts, Plans, Goals and Understanding*. Lawrence Erlbaum Associates, Hillsdale, NJ.

Schank, R.C., Fano, A., Bell, B., & Jona, K. (1993/1994). The Design of Goal-Based Scenarios. *The Journal of the Learning Sciences*.

Schank, R.C. & Leake, D.B. (1989). Creativity and Learning in a Case-Based Explainer. *Artificial Intelligence*, 40(1–3):353–385. Also appears in J.G. Carbonell, editor, *Machine Learning: Paradigms and Methods*, MIT Press, Cambridge, MA, 1990.

Schlimmer, J.C. (1987). Incremental Adjustment of Representations for Learning. In *Proceedings of the Fourth International Workshop on Machine Learning*, pages 79–90, Irvine, CA.

Seifert, C. (1988). Goals in Reminding. In J.L. Kolodner, editor, *Proceedings of the Case-Based Reasoning Workshop*, pages 190–208, Morgan Kaufmann Publishers, San Mateo, CA.

Slade, S. (1993). *Goal-Based Decision Making: An Interpersonal Model*. Lawrence Erlbaum Associates, Hillsdale, NJ.

Spellman, B.A. & Holyoak, K.J. (1993). An Inhibitory Mechanism for Goal-Directed Analogical Mapping. In *Proceedings of the Fifteenth Annual Conference of the Cognitive Science Society*, pages 947–952, Boulder, CO.

Sperber, D. & Wilson, D. (1986). *Relevance: Communication and Cognition*. Language and Thought Series, Harvard University Press, Cambridge, MA.

Srull, T. & Wyer, R. (1986). The Role of Chronic and Temporary Goals in Social Information Processing. In R. Sorrentino and E. Higgins, editors, *Handbook of Motivation and Cognition: Foundations of Social Behavior*, pages 503–549, Guilford Press, Guilford, CT.

Steinbart, P.J. (1992). The Role of Questioning in Learning from Computer-Based Decision Aids. In T.W. Lauer, E. Peacock, and A.C. Graesser, editors, *Questions and Information Systems*, pages 273–285, Lawrence Erlbaum Associates, Hillsdale, NJ.

Stroulia, E. & Goel, A.K. (1994). Task Structures: A Language for Learning Goals. In M. desJardins & A. Ram, editors, *Proceedings of the AAAI Spring Symposium on Goal-Driven Learning*, pages 112–121, Stanford, CA.

Stroulia, E., Shankar, M., Goel, A.K., & Penberthy, L. (1992). A Model-Based Approach to Blame Assignment in Design. In J.S. Gero, editor, *Proceedings of the Second International Conference on AI in Design*, pages 519–537.

Sussman, G. (1975). *A Computer Model of Skill Acquisition.* Artificial Intelligence Series, Volume 1, American Elsevier, New York, NY.

Tambe, M., Newell, A., & Rosenbloom, P.S. (1990). The Problem of Expensive Chunks and its Solution by Restricting Expressiveness. *Machine Learning*, 5:299-348.

Thagard, P. & Millgram, E. (chapter 19 of this volume). Inference to the Best Plan: A Coherence Theory of Decision. In A. Ram & D.B. Leake, editors. *Goal-Driven Learning*, MIT Press/Bradford Books, Cambridge, MA.

Utgoff, P.E. (1986). Shift of Bias for Inductive Concept Learning. In R.S. Michalski, J.G. Carbonell, & T.M. Mitchell, editors, *Machine Learning II: An Artificial Intelligence Approach*, Morgan Kaufman Publishers, San Mateo, CA.

van Berkum, J.J.A., Hijne, H., de Jong, T., van Joolingen, W.R., & Njoo, M. (1991/chapter 15). Aspects of Computer Simulations in an Instructional Context. *Education and Computing*, 6:231-239, 1991.

Van de Velde, W. (1988). Quality of Learning. In *Proceedings of the European Conference on Artificial Intelligence*, pages 408-413, Munich, Germany.

VanLehn, K. (1989). Problem Solving and Cognitive Skill Acquisition. In M.I Posner, editor, *Foundations of Cognitive Science*, pages 527-579, MIT Press, Cambridge, MA.

VanLehn, K. (1991a). Rule Acquisition Events in the Discovery of Problem Solving Strategies. *Cognitive Science*, 15(1):1-47.

VanLehn, K. (1991b). *Architectures for Intelligence: The Twenty-Second Carnegie-Mellon Symposium on Cognition.* Lawrence Erlbaum Associates, Hillsdale, NJ.

Veloso, M. & Carbonell, J.G. (1993). Derivational Analogy in PRODIGY: Automating Case Acquisition, Storage, and Utilization. *Machine Learning*, 10(3):249-278.

Weinert, F.E. (1987). Introduction and Overview: Metacognition and Motivation as Determinants of Effective Learning and Understanding. In F.E. Weinert & R.H. Kluwe, editors, *Metacognition, Motivation, and Understanding*, Lawrence Erlbaum Associates, Hillsdale, NJ.

Weintraub, M. (1991). *An Explanation-Based Approach to Assigning Credit.* Ph.D. Dissertation, Computer Science Department, The Ohio State University, Columbus, OH.

Wellman, H.M. (1985). The Origins of Metacognition. In D.L. Forrest-Pressley, G.E. MacKinnon, & T.G. Waller, editors, *Metacognition, Cognition, and Human Performance*, Volume 1, pages 1-31, Academic Press, New York.

Wellman, H.M. (1992). *The Child's Theory of Mind.* MIT Press/Bradford Books, Cambridge, MA.

White, R.T. & Gunstone, R.F. (1989). Metalearning and Conceptual Change. *International Journal of Science Education*, 11:577-586.

Wilensky, R. (1983). *Planning and Understanding.* Addison-Wesley, Reading, MA.

Wisniewski, E.J. & Medin, D.L. (1991/chapter 6 of this volume). Harpoons and Long Sticks: The Interaction of Theory and Similarity in Rule Induction. In D. Fisher & M.J. Pazzani, editors, *Concept Formation: Knowledge and Experience in Unsupervised Learning*, Morgan Kaufman Publishers, San Mateo, CA.

Wrobel, S. (1988). Automatic Representation Adjustment in an Observational Discovery System. In D. Sleeman, editor, *Proceedings of the Third European Working Session on Learning*, pages 253–261, Pitman Publishing, London.

Xia, X. & Yeung, D.-Y. (1988/chapter 12). A Learning Model for the Selection of Problem Solving Strategies in Continuous Physical Systems. In *Proceedings of the Fourth IEEE Conference on Artificial Intelligence Applications*, pages 200–206, San Diego, CA.

Zukier, H. (1986). The Paradigmatic and Narrative Modes in Goal-Guided Inference. In R. Sorrentino and E.T. Higgins, editors, *Handbook of Motivation and Cognition: Foundations of Social Behavior*, Guilford Press, New York, NY.

Proceedings of the Fourth World Conference on Engineering Education
St. Paul, Minnesota; October, 1995.

STRUCTURING ON-THE-JOB TROUBLESHOOTING PERFORMANCE TO AID LEARNING

Brian Minsk
Center for Human-Machine Systems Research
School of Industrial & Systems Engineering
Georgia Institute of Technology
Atlanta, GA 30332
E-mail: minsk@chmsr.gatech.edu

Harinarayanan Balakrishnan and Ashwin Ram
College of Computing
Georgia Institute of Technology
Atlanta Georgia 30332
E-mail: {harib,ashwin}@cc.gatech.edu

This paper describes a methodology for aiding the learning of troubleshooting tasks in the course of an engineer's work. The approach supports learning in the context of actual, on-the-job troubleshooting and, in addition, supports performance of the troubleshooting task in tandem. This approach has been implemented in a computer tool called WALTS (Workspace for Aiding and Learning Troubleshooting). This method aids learning by helping the learner structure his or her task into the conceptual components necessary for troubleshooting, giving advice about how to proceed, suggesting candidate hypotheses and solutions, and automatically retrieving cognitively relevant media. WALTS includes three major components: a structured dynamic workspace for representing knowledge about the troubleshooting process and the device being diagnosed; an intelligent agent that facilitates the troubleshooting process by offering advice; and an intelligent media retrieval tool that automatically presents candidate hypotheses and solutions, relevant cases, and various other media. WALTS creates resources for future learning and aiding of troubleshooting by storing completed troubleshooting instances in a self-populating database of troubleshooting cases. The methodology described in this paper is partly based on research in problem-based learning, learning by doing, case-based reasoning, intelligent tutoring systems, and the transition from novice to expert. The tool is currently implemented in the domain of remote computer troubleshooting.

Introduction

Sophisticated devices and systems, such as computers, are widespread and growing in use. With the growth of complex technology an imperative need for efficient and effective troubleshooting skills has developed. Troubleshooting technologically sophisticated devices is often a difficult, complex process to carry out and learn: efficient, effective troubleshooting generally requires intimate knowledge of complicated devices and proficient reasoning skills and strategies.

Computing technology has advanced to the point where tools supporting both the performance and learning of complex engineering tasks such as troubleshooting are now possible. Many troubleshooters employ advanced computing technology in the course of their jobs. These computers are often used to help troubleshooters *carry out* their jobs, but the same computers are typically not used to help troubleshooters *learn* to do their jobs better. While these computers could be used to provide training separate from the performance of troubleshooting, a more effective approach may be to use these computers to support learning "on-the-job," while professionals are engaged in actual troubleshooting. Learning from direct experience, or "learning by doing," has been acknowledged as an effective educational approach [6].

This paper describes a methodology for aiding the learning of troubleshooting tasks in the course of a professional engineer's work. This methodology has been realized in a computer tool called WALTS (Workspace for Aiding and Learning TroubleShooting). WALTS is currently implemented in the domain of remote computer troubleshooting. Each component of this methodology and its implementation in the WALTS tool is described in the following sections. The Summary section provides a synopsis of the methodology involved in producing WALTS and its relation to previous work.

Structuring Troubleshooting to Aid Learning

Because of the complexity and constant innovation of sophisticated technology, professional troubleshooters must perpetually be learning about their domain. To become more efficient and effective performers, troubleshooters must also continually add to and refine their reasoning skills and strategies. A particular challenge for aiding on-the-job learning is to support learning without harming performance. Ideally, such a methodology should produce benefits for both performance and learning. The methodology described in this paper is based on structuring the performance of troubleshooting such that both performance and learning are enhanced in tandem.

The WALTS system includes three major components for structuring troubleshooting (see Figure 1). One, the Cognitive Troubleshooting Workspace and the Causal Links Workspace allows troubleshooters to represent knowledge about the troubleshooting process and the device being diagnosed. Next, the Facilitator is an intelligent agent that facilitates the troubleshooting process by offering advice. Finally, an intelligent media retrieval tool called the Media Retrieval Workspace automatically presents candidate hypotheses and solutions, relevant cases, and various other media.

Representing the Conceptual Structure of Troubleshooting

The methodology described in this paper involves representing knowledge in a form that is functional for troubleshooting. Representing troubleshooting knowledge entails separating the knowledge used in troubleshooting into conceptual components, specifying supporting and disconfirming relations between the components, and specifying causal relations between the components. In the Cognitive Troubleshooting Workspace of the WALTS system, a troubleshooter can document, structure, classify and display the knowledge used in the troubleshooting and create and display supporting or disconfirming relationships between objects. In the Causal Links Workspace troubleshooters can create and display causal relations between objects.

The knowledge used in the troubleshooting process is separated into its conceptual components. Five conceptual components are identified and described below: symptoms, configuration data, state data, hypotheses, tests, and solutions.

- *Symptoms.* Symptoms are the set of system behaviors someone considers problematic that the troubleshooter is trying to correct and, perhaps, find causes for. Example symptoms for a computer system might include the system's inability to make a network connection or format a disk. For a human, symptoms might include a rash or chest pains.

- *Configuration data*. Configuration data includes the relatively stable parts of a system. For a computer system this might include hardware and software. For a human, this might include sex, weight, or age.
- *State data*. State data is the set of relatively inconstant system behaviors (not symptoms) and the settings or values of changeable system variables. Examples for a computer system might include a condition when a computer is connecting to some devices on a network but not others or the setting of some variable in a configuration file. Examples for a human might include heart rate, blood pressure, or blood alcohol level.
- *Hypotheses*. Hypotheses are the possible causes of a set of symptoms. For a computer, this might be a broken piece of hardware or a wrong setting of some system variable. For a human, this might be a tumor, disease, or dietary problem.
- *Tests*. Tests are the actions taken to demonstrate the validity of a hypothesis or solution. Tests can be implemented by changing the value of a system variable or adding or subtracting components from the system. The result is a set of (hopefully) observable system behaviors. If a troubleshooter suspects a network card is the cause of a symptom in a computer, a test might include removing the network card to see if the symptom behavior is still displayed by the computer. An example test on a human might include implementing a low salt diet to validate a hypothesis that high sodium levels are the cause of an individual's high blood pressure.
- *Solutions*. Solutions are the actions taken to correct a set of symptoms. Solutions can be implemented by changing the value of some system. A solution may be the same as a test: however, solutions are specific to changing the behaviors of the system that are expressed as the set of symptoms. An example solution for a computer system includes replacing a bad network card when a computer system is not connecting to the network. A solution for a human condition might include removing a malignant brain tumor to restore normal brain function.

These conceptual components explicitly structure the knowledge used to carry out the troubleshooting task in a way that supports the development and evaluation of hypotheses and solutions. Each fragment of knowledge used in the troubleshooting process is classified with these conceptual components into "knowledge objects". An explicit representation of the knowledge and process for troubleshooting supports learning [4].

To evaluate hypotheses and solutions in this methodology, "supporting" or "disconfirming" relations and causal relations can be established between the knowledge objects used for troubleshooting. A supporting relation indicates that a knowledge object is confirming evidence for a hypothesis or solution. Similarly, a disconfirming relation denotes negative evidence for a hypothesis or solution. With supporting and disconfirming relations between knowledge objects, a troubleshooter links and assesses hypotheses and solutions directly with system data. Causal relations between knowledge objects are used to represent the causal chains involved in generating symptoms. Understanding the causes of symptoms is important for generating hypotheses and solutions and for developing causal knowledge about a device.

Troubleshooters can create and classify their knowledge by creating text windows in the Cognitive Troubleshooting Workspace. Each text window identifies the type of knowledge object it is in the title bar, the knowledge itself (typed in by the troubleshooter), and any supporting or disconfirming relationship with an active text window. Supporting and disconfirming relations are created by dragging and dropping between text windows and specifying the type of relation in a dialog box. The bar at the bottom of the text window is displayed as green if it has a supporting relationship with the active window, red if disconfirming, and gray if no relation has been specified by a troubleshooter.

Troubleshooters can use another workspace in WALTS called the Causal Links Workspace to create causal links between objects. Users drag objects from the Cognitive Troubleshooting Workspace into the Causal Links Workspace. In the Causal Links Workspace users can drag and drop objects to other objects to create causal relations that are represented by arrows.

Intelligently Facilitating the Troubleshooting Process

Learning reasoning skills and strategies involved in troubleshooting complex devices is also supported by "facilitating" troubleshooters' reasoning to make it more efficient and effective. Facilitating involves actively offering timely advice while troubleshooters carry out the troubleshooting task. For example, if a troubleshooter generates a large number of hypotheses, advice to try to eliminate hypotheses or to evaluate the most likely hypotheses is given immediately. Or, if a troubleshooter tries to implement a solution that has disconfirming evidence, the system suggests the troubleshooter avoid a solution that has negative evidence.

Advice can and sometimes should be ignored by troubleshooters. As troubleshooters learn to make their reasoning more effective and efficient, advice should become less obtrusive. In the present methodology troubleshooters can control how obtrusive the advice is such that it can go from being totally intrusive to the troubleshooting process to providing no advice at all.

In the WALTS system, advice about how to make the troubleshooting process effective and efficient is given by an intelligent agent called the Facilitator. The Facilitator actively observes the Cognitive Troubleshooting Workspace. When it encounters a condition in which it can give advice, it offers this advice to the user. Advice to the user can be presented with a beep, with a flash of the screen, in a dialog box that the user must clear from the screen, in a simple text window (as displayed in Figure 1), not at all, or any combination of the above as controlled by users.

Intelligently Retrieving Relevant Information

The methodology described in this paper embodies four major goals for providing information for the learning troubleshooter. First, information furnished by the tool to troubleshooters serves their specific problem-solving and learning goals [4]. Next, access to manuals, technical specifications, instructions, and past troubleshooting cases is provided and involves little or no effort on the part of the troubleshooter. Third, retrieved media are labeled with a "cognitive media type" which classifies the media's potential utility for problem-solving and learning [5]. Finally, hypotheses and solutions based on past troubleshooting cases are suggested when appropriate.

The Media Retrieval Workspace in WALTS provides the troubleshooter's interface to media retrieval. The Media Retrieval Workspace automatically retrieves media based on the current contents of the Cognitive Troubleshooting Workspace. Users can also initiate manual searches in the Media Retrieval Workspace. WALTS has access to large free-form text databases of previously completed troubleshooting cases relevant to its domain and various multimedia, on-line manuals, instructions, and technical specifications. WALTS also has access to previous troubleshooting cases accomplished with the WALTS tool. Retrieved media is labeled with its cognitive media type, its physical media type (e.g. text, picture, sound or movie), the type of WALTS object or the database from which it comes, a short description, and a relevance metric.

The structured knowledge furnished by the Cognitive Troubleshooting Workspace allows for efficient, directed information retrieval and storage. The Media Retrieval Workspace's automatic retrieval uses the structure of the knowledge contained in the Cognitive Troubleshooting Workspace to weight and prune searches. These directed searches help overcome a traditional problem of retrieving more media from a large database than a user can handle. Each retrieved media includes a display of the Cognitive Troubleshooting Workspace categories from which it was retrieved. Relevance metrics are computed and displayed to represent the quality of the matched media. In addition, previous WALTS cases are stored in a structured format allowing for much more efficient and useful access than is normally allowed with free-form text databases. If a Hypothesis or Solution is retrieved from a previous WALTS case, the user can simply double-click on the the retrieved object in the Media Retrieval Workspace to add it to the Cognitive Troubleshooting Workspace. Users can also display a complete previous WALTS case retrieved with the Media Retrieval Workspace.

Summary

In the domain of WALTS, the troubleshooters' learning goals are to acquire knowledge about the devices being diagnosed and to improve reasoning skills and strategies associated with troubleshooting. The problem-solving goals are to generate a successful explanation (hypothesis) and remedy (solution) for a device's problems. The methodology described in this paper attempts to provide support for both sets of goals in tandem by providing a structured workspace to document the troubleshooting process, an intelligent "facilitator" with troubleshooter-controlled obtrusiveness to make the process more effective and efficient, and automatically furnished information that is useful and relevant.

The emphasis in the methodology used to produce WALTS is to structure the "doing" of troubleshooting such that performance and learning of troubleshooting is enhanced. Expert problem-solvers represent knowledge according to abstract principles relevant for problem-solving [2]. With the structure of the Cognitive Troubleshooting Workspace, WALTS attempts to "jump-start" this type of abstract classification for novices. Like the SHERLOCK system [3], WALTS provides an intelligent facilitation environment for learning anchored in experience. Unlike SHERLOCK, in which the training is separated from performance, WALTS is embedded in the actual performance situation. As in problem-based learning environments, the methodology described in this paper grounds knowledge about the devices being diagnosed in the performance situation itself and supports the development of diagnostic reasoning skills [1].

Acknowledgments

This research was supported by a grant from the Digital Equipment Corporation and by the Edutech Institute. The authors thank Greg Heathcock, Chris Hill, and Anil Rewari for their contributions to this research and Alan Chappell and Jennifer Turns for their helpful comments on an earlier draft of this manuscript.

References

[1] Barrows, H. S. (1986). A taxonomy of problem-based learning methods. *Medical Education, 20.*

[2] Chi, M. T. H., Feltovich, P. J., & Glaser, R. (1981). Categorization and representation of physics problems by experts and novices. *Cognitive Science, 5.*

[3] Lajoie, S. P. & A. Lesgold (1989). Apprenticeship training in the workplace: computer-coached practice environment as a new form of apprenticeship. *Machine-Mediated Learning, 3.*

[4] Ram, A., S. Narayanan, & M. T. Cox (1995). Learning to troubleshoot: multistrategy learning of diagnostic knowledge for a real-world problem solving task. *Cognitive Science, 19.*

[5] Recker, M. M., A. Ram, T. Shikano, G. Li, & J. Stasko (In press). Cognitive media types for multimedia information access. *Journal of Educational Multimedia and Hypermedia*.

[6] Soloway, E., M. Guzdial, & K. E. Hay (1994). Learner-centered design: the challenge for HCI in the 21st century. *Interactions*, April, Association of Computing Machinery.

To appear in *Proceedings of the Second European Workshop on Case-Based Reasoning*. Chantilly, France. Nov 7-10, 1994.

Managing Learning Goals in Strategy-Selection Problems

Michael T. Cox and Ashwin Ram
College of Computing
Georgia Institute of Technology
Atlanta, GA 30332-0280
{cox,ashwin}@cc.gatech.edu

Abstract

In case-based reasoning systems, several learning techniques may apply to a given situation. In a failure-driven learning environment, the problems of *strategy selection* are to choose the best set of learning algorithms or strategies that recover from a processing failure and to use the strategies to modify the system's background knowledge so that the failure will not repeat in similar future situations. A solution to this problem is to treat learning-strategy selection as a planning problem with its own set of goals. Learning goals, as opposed to ordinary goals, specify desired states in the background knowledge of the learner, rather than desired states in the external environment of the planner. But as with traditional goal-based planners, management and pursuit of these learning goals becomes a central issue in learning. Examples are presented from a multistrategy learning system called Meta-AQUA that combines a case-based approach to learning with nonlinear planning in the knowledge space.

1 Introduction

As case-based reasoning (CBR) research addresses more sophisticated task domains, the associated learning issues involved become increasingly complex. Multistrategy learning systems attempt to address the complexity of such task domains by bringing to bear many of the learning algorithms developed in the last twenty years. Yet the goal of integrating various learning strategies is a daunting one, since it is an open question as how best to combine often conflicting learning-mechanisms. This paper examines the metaphor of goal-driven problem-solving as a tool for performing this integration. Learning is viewed as simply another problem to solve; the learning problem is formulated by posting learning goals (such as goals to answer a question or to reconcile two divergent assertions). A plan is assembled by choosing various learning algorithms from the system's repertoire that can achieve such goals.

In formulations with conjunctive goals, however, numerous difficulties arise such as goal conflicts, protection intervals, and many-to-many relationships between goals and algorithms. For instance, a familiar goal conflict in planning systems is the "brother-clobbers-brother" goal interaction (Sussman, 1975), whereby the result of one plan that achieves a particular goal undoes the result or precondition of another plan serving a different goal. If a learning goal specifies a state change to the background knowledge of the system, rather than a state change in the world, then learning plans can have similar effects. Changes to specific knowledge may affect previous changes to the background knowledge performed by other learning algorithms.

This paper illustrates such interactions with a multistrategy learning system called Meta-AQUA. The performance task in Meta-AQUA is story understanding. That is, given a stream of concepts as the representation for a story sequence, the task is to create a causally connected conceptual interpretation of the story. The story-understanding strategies available to the system are CBR, analogy, and explanation. If the system fails at its performance task, its subsequent learning-subtasks are (1) *blame assignment* — use case-based methodologies to analyze and explain the cause of its misunderstanding by retrieving past cases of meta-reasoning, (2) *decide what to learn* — use these cases to deliberately form a set of learning goals to change its knowledge so that such a misunderstanding is not repeated on similar stories, and then (3) *strategy selection* — use nonlinear planning techniques to choose or construct some learning method by which it achieves these goals. These stages are detailed in Figure 1.

Previous publications have dealt with the blame assignment stage (Cox, 1993; Cox & Ram, 1992; Ram & Cox, 1994). This paper explores how learning goals are spawned when deciding what to learn (Section 2) and how these goals are satisfied in the strategy-selection phase (Section 3). A simpler system might forego explicit learning-goals altogether, and directly map a failure to a learning algorithm. The discussion of Section 3 explores, not only how learning goals are managed, but what leverage is gained over and above a direct mapping itself. The paper's conclusion (Section 4) specifies the relation between the Meta-AQUA system and traditional CBR approaches and discusses related systems, limitations with Meta-AQUA, and areas for future research.

```
0. Perform and Record Reasoning in Trace
1. Failure Detection on Reasoning Trace
2. If Failure Then
      Learn from Mistake:
        • 2a. Blame Assignment
            Compute index as characterization of failure
            Retrieve Meta-XP
            Apply Meta-XP to trace of reasoning
            If XP application is successful then
              Check XP antecedents
              If one or more nodes not believed then
                Introspective questioning
                GOTO step 0
              Else GOTO step 0
        • 2 b. Create Learning Goals
            Compute tentative goal priorities
        • 2 c. Choose Learning Algorithm(s)
            Translate Meta-XP and goals to predicates
            Pass goals and Meta-XP to Nonlin
            Translate resultant plan into frames
        • 2 d. Apply Learning Algorithm(s)
            Interpret plan as partially ordered network of
                actions such that primitive actions are
                algorithm calls
3. Evaluate Learning (not implemented)
```

Figure 1: Meta-AQUA's learning algorithm

S1: A police dog sniffed at a passenger's luggage in the Atlanta airport terminal.

S2: The dog suddenly began to bark at the luggage.

S3: The authorities arrested the passenger, charging him with smuggling drugs.

S4: The dog barked because it detected two kilograms of marijuana in the luggage.

Figure 2: The drug-bust story

2 Deciding What to Learn

Learning goals represent what a system needs to know (Ram, 1991; 1993; Ram & Hunter, 1992; Ram & Leake, in press) and are spawned when deciding what to learn. Learning goals help guide the learning process by suggesting strategies that would allow the system to learn the required knowledge.[1] Given some failure of a reasoner, the task of the learning system is to adjust its knowledge so that such reasoning failures will not recur in similar situations.[2] The learner is therefore modeled as a planning system that spawns goals to achieve this overall task (Hunter, 1990; Ram & Hunter, 1992). The learner subsequently attempts to create plans resulting in desired new states of its background knowledge[3] that satisfy these goals. The overall aim is to turn reasoning failures into opportunities to learn and to improve the system's performance.

1. Learning goals also facilitate opportunistic learning (see Ram, 1991; 1993; Ram & Hunter, 1992), that is, if all information necessary for learning is not available at the time it is determined what is needed to be learned (e.g., when a question is posed), then a learning goal can be suspended, indexed in memory, and resumed at a later time when the information becomes available.

2. The learner could also adjust its circumstances in the physical world, such as placing items in a cupboard in the same place to aid memory retrieval. This paper, however, will not entertain such possibilities. See Hammond (1990) for an approach to such task interactions and associated learning.

3. The background knowledge includes more than simple domain knowledge. It can also contain knowledge such as metaknowledge, heuristic knowledge, associative knowledge, and knowledge of process.

Learning goals deal with the structure and content of knowledge, as well as the ways in which knowledge is organized in memory. Some learning goals seek to add, delete, generalize or specialize a given concept or procedure. Others deal with the ontology of the knowledge, i.e., with the kinds of categories that constitute particular concepts. Many learning goals are unary in that they take a single target as argument. For example, a *knowledge acquisition goal* seeks to determine a single piece of missing knowledge, such as the answer to a particular question. A *knowledge refinement goal* seeks a more specialized interpretation for a given concept in memory, whereas a *knowledge expansion goal* seeks a broader interpretation that explores connections with related concepts. Other learning goals are binary in nature since they take two arguments. A *knowledge differentiation goal* is a goal to determine a change in a body of knowledge such that two items are separated conceptually. In contrast, a *knowledge reconciliation goal* is one that seeks to merge two items that were mistakenly considered separate entities. Both expansion goals and reconciliation goals may include/spawn a *knowledge organization goal* that seeks to reorganize the existing knowledge so that it is made available to the reasoner at the appropriate time, as well as modify the structure or content of a concept itself. Such reorganization of knowledge affects the conditions under which a particular piece of knowledge is retrieved or the kinds of indexes associated with an item in memory.

A program called Meta-AQUA (Ram & Cox, 1994) was written to test our theory of understanding, explanation and learning. Given the drug-bust story of Figure 2, the system attempts to understand each sentence by incorporating it into its current story representation, explain any anomalous or interesting features of the story, and learn from any reasoning failures. Numerous inferences can be made from this story, many of which may be incorrect or incomplete, depending on the knowledge of the reader. Meta-AQUA's background knowledge includes general facts about dogs and sniffing, including the fact that dogs bark when threatened, but it has no knowledge of police dogs. It also knows cases of gun smuggling, but has never seen drug interdiction. The learning task in Meta-AQUA is to learn from failures, incrementally improving its ability to interpret new stories.

In the drug-bust story, sentence S1 produces no inferences other than that sniffing is a normal event in the life of a dog. However, S2 produces an anomaly because the system's definition of "bark" specifies that the object of a bark is animate. So the program (incorrectly) believes that dogs bark only when threatened by animate objects (see Figure 3 for the representation[4] produced by Meta-AQUA during blame assignment). Since luggage is inanimate, there is a contradiction, leading to an incorporation failure. This anomaly causes the understander to ask why the dog barked at an inanimate object. It is able to produce but one explanation: the luggage somehow threatened the dog.

S3 asserts an arrest scene which reminds Meta-AQUA of a past case of weapons smuggling by terrorists; however, the sentence generates no new inferences concerning the previous anomaly. Finally, S4 causes the question generated by S2, "Why did the dog bark at the luggage?" to be retrieved. Instead of revealing the anticipated threatening situation, S4 offers another hypothesis: "The dog detected drugs in the luggage."

The system characterizes the reasoning error as an expectation failure caused by the incorrect retrieval of a known explanation ("dogs bark when threatened by objects," erroneously assumed to be applicable), and a missing explanation ("the dog barked because it detected marijuana," the correct explanation in this case). During blame assignment, Meta-AQUA uses this characterization as an index to retrieve an abstract case called a Meta-XP (Ram & Cox, 1994) that is applied to a trace of the reasoning that produced the failure. The instantiation results in an explanation of its reasoning error, as shown in Figure 3. This composite meta-explanation consists of three parts: a Novel-Situation centered about "Retrieval Failure," an Erroneous-Association centered about "Expectation Failure" and an Incorrect-Domain-Knowledge centered about "Incorporation Failure."

Faced with the structure of the reasoning error produced by the blame-assignment phase, the learner determines the learning goals for the system. First, since the seemingly anomalous input (marked "Old Input" in Figure 3) has been incorporated in the story and later reinforced by the coherence of the story structure, and since no contradictions occurred as a result of this inference, the learner posts a knowledge reconciliation goal. The goal is to adjust the background knowledge so that neither dogs barking at animate objects nor dogs barking at inanimate objects will be considered anomalous by the understander. This learning goal is appropriate because even though one item is an

4. Attributes and relations are represented explicitly in Meta-AQUA and in this figure. For instance, the ACTOR attribute of an event Dog-bark.12 with the value Dog.4 is equivalent to the explicitly represented relation ACTOR.21 having a `domain` value of Dog-bark.12 and a `co-domain` value of Dog.4. In addition, all references to TRUTH attributes equal to `out` refer to the `domain` being out of the current set of beliefs. See Cox (1993) and Ram & Cox (1994) for further representational details. The "Internal Structures Window" of Figure 4 shows the top-level frame representation corresponding to Figure 3.

instantiated token (a particular dog barked at a specific inanimate object), while the other is a type definition (concept specifying that dogs generally bark at animate objects), they are similar enough to each other to be reconcilable.

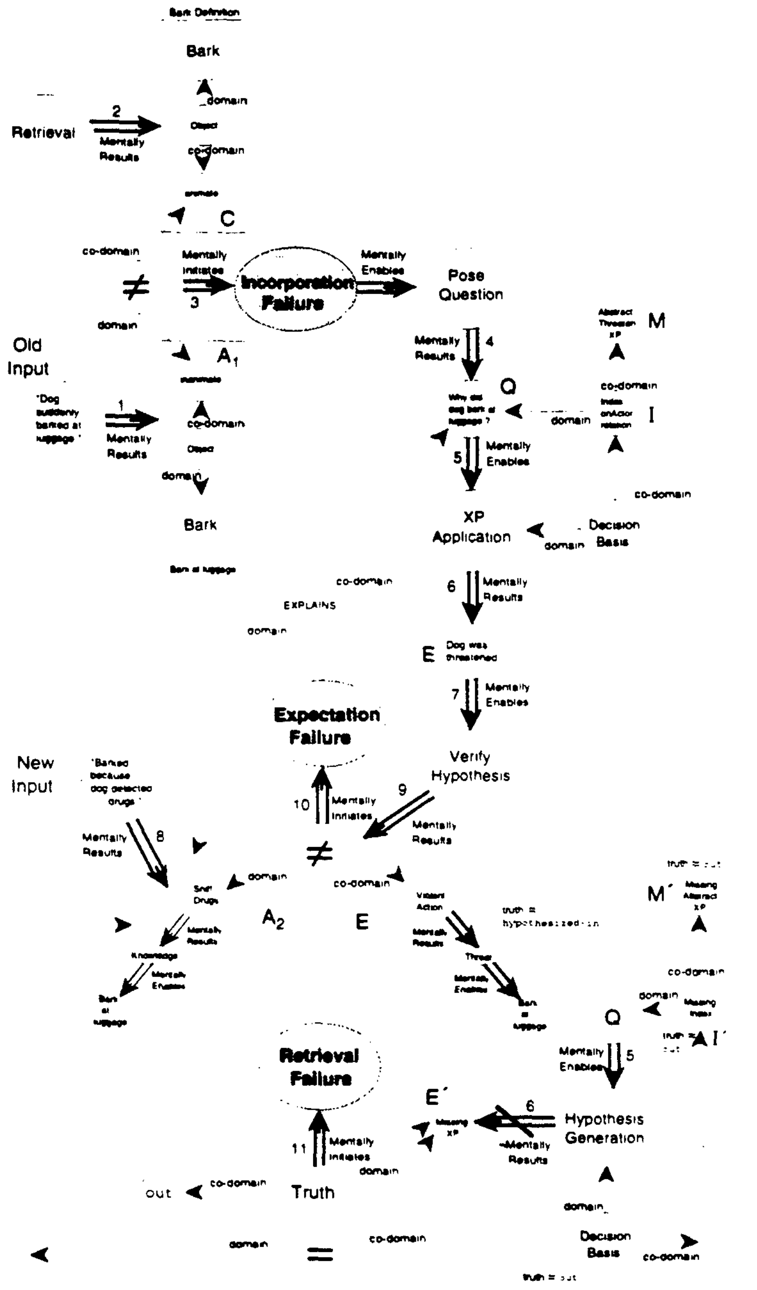
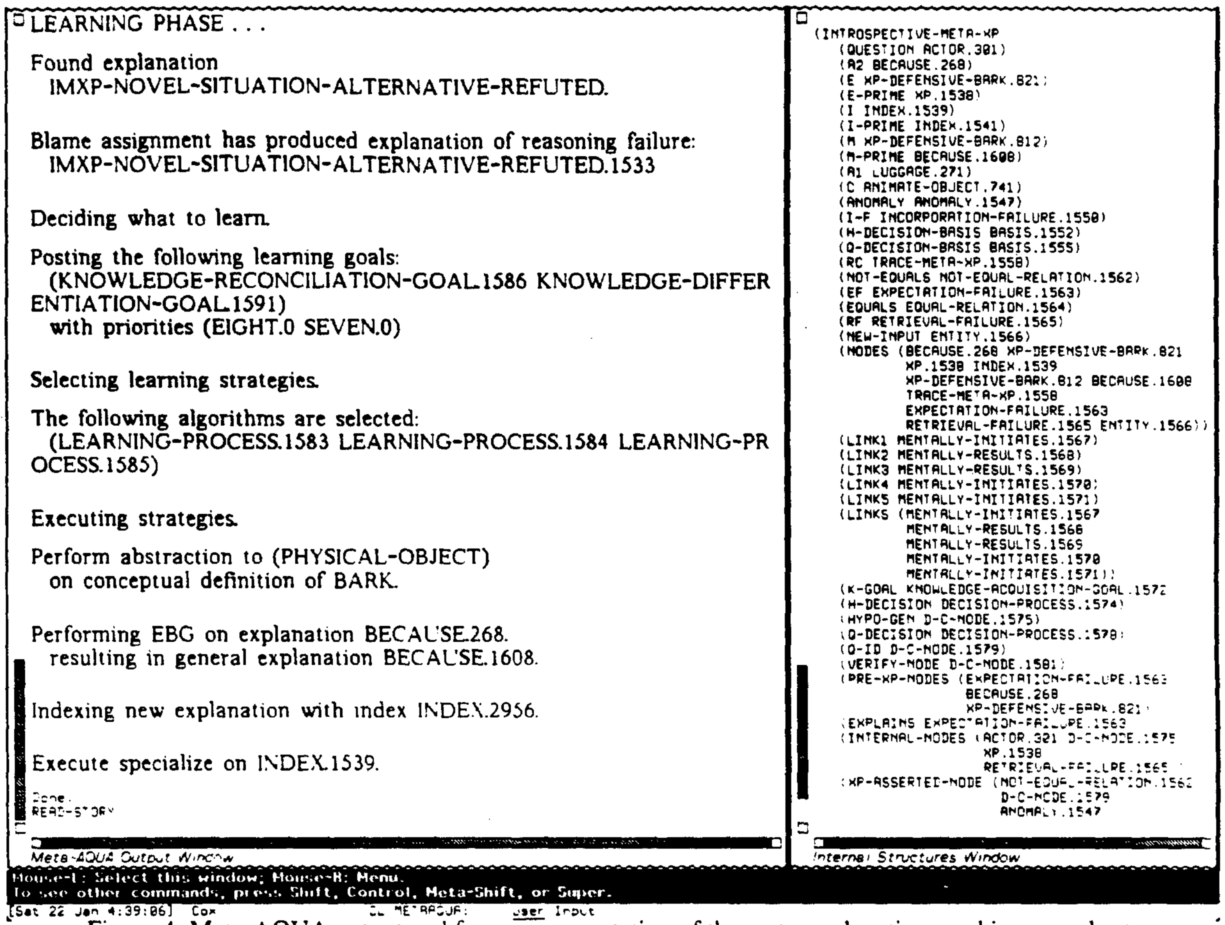


Figure 3: Instantiated composite meta-explanation

Secondly, given that an expectation failure triggered the learning, and (from the blame assignment phase) given that the failure resulted from the interaction of misindexed knowledge and a novel situation, Meta-AQUA posts a goal to differentiate between the two explanations for why the dog barked (nodes M and M′ in Figure 3). Since the conflicting explanations are significantly different (for example, they do not share the same predicate, i.e., detect versus threaten), a knowledge-differentiation goal is licensed, rather than a goal to reconcile the two types of explanations. The differentiation goal is achieved if the system can retrieve proper explanations given various situations. The original misunderstanding of the story occurred, not because the explanation that dogs bark when threatened is incorrect in general, but rather because the system did not know the proper conditions under which this explanation applies.



```
LEARNING PHASE . . .

  Found explanation
    IMXP-NOVEL-SITUATION-ALTERNATIVE-REFUTED.

  Blame assignment has produced explanation of reasoning failure:
    IMXP-NOVEL-SITUATION-ALTERNATIVE-REFUTED.1533

  Deciding what to learn.

  Posting the following learning goals:
    (KNOWLEDGE-RECONCILIATION-GOAL.1586 KNOWLEDGE-DIFFER
  ENTIATION-GOAL.1591)
    with priorities (EIGHT.0 SEVEN.0)

  Selecting learning strategies.

  The following algorithms are selected:
    (LEARNING-PROCESS.1583 LEARNING-PROCESS.1584 LEARNING-PR
  OCESS.1585)

  Executing strategies.

  Perform abstraction to (PHYSICAL-OBJECT)
    on conceptual definition of BARK.

  Performing EBG on explanation BECAUSE.268.
    resulting in general explanation BECAUSE.1608.

  Indexing new explanation with index INDEX.2956.

  Execute specialize on INDEX.1539.

  Done.
  READ-STORY

Meta-AQUA Output Window
```

```
(INTROSPECTIVE-META-XP
   (QUESTION ACTOR.301)
   (A2 BECAUSE.268)
   (E XP-DEFENSIVE-BARK.821)
   (E-PRIME XP.1538)
   (I INDEX.1539)
   (I-PRIME INDEX.1541)
   (M XP-DEFENSIVE-BARK.812)
   (M-PRIME BECAUSE.1608)
   (A1 LUGGAGE.271)
   (C ANIMATE-OBJECT.741)
   (ANOMALY ANOMALY.1547)
   (I-F INCORPORATION-FAILURE.1550)
   (H-DECISION-BASIS BASIS.1552)
   (Q-DECISION-BASIS BASIS.1555)
   (RC TRACE-META-XP.1558)
   (NOT-EQUALS NOT-EQUAL-RELATION.1562)
   (EF EXPECTATION-FAILURE.1563)
   (EQUALS EQUAL-RELATION.1564)
   (RF RETRIEVAL-FAILURE.1565)
   (NEW-INPUT ENTITY.1566)
   (NODES (BECAUSE.268 XP-DEFENSIVE-BARK.821
           XP.1538 INDEX.1539
           XP-DEFENSIVE-BARK.812 BECAUSE.1608
           TRACE-META-XP.1558
           EXPECTATION-FAILURE.1563
           RETRIEVAL-FAILURE.1565 ENTITY.1566))
   (LINK1 MENTALLY-INITIATES.1567)
   (LINK2 MENTALLY-RESULTS.1568)
   (LINK3 MENTALLY-RESULTS.1569)
   (LINK4 MENTALLY-INITIATES.1570)
   (LINK5 MENTALLY-INITIATES.1571)
   (LINKS (MENTALLY-INITIATES.1567
           MENTALLY-RESULTS.1566
           MENTALLY-RESULTS.1569
           MENTALLY-INITIATES.1570
           MENTALLY-INITIATES.1571))
   (K-GOAL KNOWLEDGE-ACQUISITION-GOAL.1572
   (H-DECISION DECISION-PROCESS.1574)
   (HYPO-GEN D-C-NODE.1575)
   (Q-DECISION DECISION-PROCESS.1578)
   (Q-ID D-C-NODE.1579)
   (VERIFY-NODE D-C-NODE.1581)
   (PRE-XP-NODES (EXPECTATION-FAILURE.1563
                  BECAUSE.268
                  XP-DEFENSIVE-BARK.821)
   (EXPLAINS EXPECTATION-FAILURE.1563
   (INTERNAL-NODES (ACTOR.301 D-C-NODE.1575
                    XP.1538
                    RETRIEVAL-FAILURE.1565)
   (XP-ASSERTED-NODE (NOT-EQUAL-RELATION.1562
                      D-C-NODE.1579
                      ANOMALY.1547

Internal Structures Window
```

```
Mouse-L: Select this window; Mouse-R: Menu.
To see other commands, press Shift, Control, Meta-Shift, or Super.
[Sat 22 Jan 4:39:06]  Cox                CL METAAQUA:     User Input
```

Figure 4: Meta-AQUA output and frame representation of the meta-explanation used in example story

In addition to posting these two learning goals. Meta-AQUA places a tentative ordering on their execution (see Figure 4 for program output when deciding what to learn). With no other specific knowledge concerning their respective relations. a good default heuristic is to order them by temporal order of the failures involved in the original reasoning trace. The reason this may be useful is that if it is determined that the first failure was indeed not an error but a misunderstanding or was caused by faulty input. then the reasoning that followed from the first failure (or other assumptions depending on the nature of the first failure that led to the second) may have contributed to the cause of the second. Thus. learning acquired from the first failure may show that the subsequent reasoning was irrelevant. or it may yield more information to be used on the second goal. Therefore. the decide-what-to-learn stage outputs the knowledge-reconciliation goal with priority over the knowledge-differentiation goal.

3 Strategy Selection and Combination

In the strategy-selection stage, Meta-AQUA constructs a learning plan to realize the learning goals posted by the previous stage (see Figure 4 for program output during strategy selection phase). This entails not only choosing the algorithms and operators to achieve the learning goals. but perhaps also spawning new subgoals. To help the system create a plan for a learning goal with two arguments. the following types of questions can be posed about the reasoning chain. For example. with the knowledge-differentiation goal. the focus starts at the error that triggered the introspective-explanation episode. that is. at the expectation failure. Given that the reasoner expected explanation E to be correct. but later decides that A_2 is the actual explanation. the system needs to determine:

- Was the actual occurrence. A_2. foreseeable?
- If so, was A_2 considered?
- Is there still a possibility that A_2 is incorrect?

- Is there still a possibility that E is correct?
- How confident is the system in A_2 or any of the input associated with establishing A_2?
- How much experience does the system have with A_2 and E's abstract progenitors?

The answers to these questions enable the system to choose learning algorithms, strategies, or operators. For example, since the explanation provided by the story (A_2) provides more coherence to the understanding of the story, the system assumes that there is no error in the input. However, because the system has no prior experience with the instance (and thus the system neither foresaw nor considered the explanation), the learner posts another goal to expand the concept, thus producing M'. Explanation-based generalization (EBG) (DeJong & Mooney, 1986; Mitchell, Keller & Kedar-Cabelli, 1986) can then be selected as an appropriate learning algorithm.

A more difficult problem is to differentiate the applicability conditions for the two generalized explanations (M', the one produced by generalizing the detection explanation, A_2, and M, the original abstract pattern that produced the initial threaten explanation, E) by modifying the indexes (I' and I) with which the system retrieves those explanations. If the two problems of erroneous association and novel situation were to be treated independently, rather than as a problem of interaction, then an indexing algorithm would not be able to ensure that the two explanations would remain distinct in the future. That is, if the learner simply detects a novel situation and automatically generalizes it, then indexes it by the salient or causal features in the explanation, and if the learner independently detects an erroneous retrieval, and re-indexes it so that the same context will not retrieve it in the future, then there is no guarantee that the resultant indexes will be mutually exclusive. Instead, the system must re-index E with respect to A_2, not simply with respect to the condition with which E was retrieved. Therefore, a direct mapping from blame assignment to strategy selection without the mediation of learning goals is problematic.

The problems to be solved, then, are determining the difference between A_2 and E, and, in the light of such differences, computing the minimal specialization of the index of E and the maximally general index of A_2 so they will be retrieved separately in the future. In the case of the story above, the problem is somewhat simplified. The difference is that retrieval based on the actor relation of barking actions (dogs) is too general. The threaten explanation applies when dogs bark at animate objects, while the detection explanation is appropriate when dogs bark at containers.

The knowledge-reconciliation goal between the conceptual definition of dog-barking being limited to animate objects and the fact that a particular dog barked at a piece of luggage can be thought of as a simple request for similarity-based learning (SBL) or inductive learning (for example, UNIMEM's SBL algorithm in Lebowitz, 1987, or abstraction transmutation as in Michalski, 1994). The system is simply adding an additional positive example to the instances seen. An incremental algorithm is required because this instance has been discovered after an initial concept has been established some time in the past.

An interesting interaction can occur, however, if the system waits for the result of the EBG algorithm required by the knowledge-expansion subgoal spawned by the knowledge-differentiation goal discussed above. The algorithm will generalize the explanation (that this particular dog barked at a particular piece of luggage because it detected marijuana) to a broader explanation (that dogs in general may bark at any container when they detect contraband). Thus, the example provided to the inductive algorithm can be more widely interpreted, perhaps allowing its inductive bias to generalize the constraint, C, on the object of dog-barking to `physical-object` (the exhaustive case of `animate-object` and `inanimate-object`), whereas a single instance of a particular breed of dog barking at a specific brand of luggage, A_1, may limit the inductive inference if no additional domain knowledge is available.

Unfortunately, however, because the EBG algorithm uses the representation of the dog-bark definition, and the inductive algorithm changes this definition, the induction must occur first. Thus, the learner cannot take advantage of the opportunity cited in the previous paragraph. One important implication of this point is that in systems which plan to learn, if the reasoner does not anticipate this second interaction (thus placing EBG before the induction), the system must be able to perform dynamic backtracking on its decisions.

To notice these types of interactions, however, requires a least-commitment approach such as that used in a non-linear hierarchical planner like Nonlin (Ghosh, Hendler, Kambhampati, & Kettler, 1992; Tate, 1976). Likewise, the system must detect any dependency relationships so that goal violations can be avoided. For example, when the definition of dog-barking is modified by generalizing its constraint of what dogs bark at to `physical-object` from `animate-object`, any indexing based on the modified attribute must occur after this modification, rather than before it, to avoid indexing with obsolete conceptual knowledge.[5]

Figure 5 shows a learning-operator definition for the indexing strategy that manages mutual indexing between two concepts. The operator schema determines that both items must be independently indexed before they are indexed with respect to each other. The action schema has filter conditions that apply when both are indexed and both are XPs. An unsupervised condition specifies that if there exists a change in the explained action, then it must occur before the execution of this schema. That is, a linearization must be performed on external goals to reorder any other schema that may establish the change. It says in effect that we want all attributes of the target concept to be stable before it operates on the concept; no other operator can change an attribute in order for the changes performed by indexing to be unaffected. Note that the action schema of abstraction in Figure 6 has an effect that includes such a change to its addlist. Therefore, if both schemas are being instantiated, Nonlin will automatically order the abstraction before the indexing. A similar unsupervised condition prevents generalization from occurring before the concept is stable.

```
(opschema mutual-index-op
        :todo   (index-wrt-item ?x ?y)
        :expansion (
                (step1 :goal (indexed ?x))
                (step2 :goal (indexed ?y))
                (step3 :action
                        (index-dual-items ?x ?y)))
        :orderings(
                (step1 -> step3)
                (step2 -> step3))
        :conditions (
                (:precond (indexed ?x)
                 :at step3 :from step1)
                (:precond (indexed ?y)
                 :at step3 :from step2)
                (:use-when (not (equal ?x ?y))
                        :at step1))
        :effects    ()
        :variables (?x ?y))
```

```
(actschema do-mutual-xp-indexing
        :todo   (index-dual-items ?x ?y)
        :expansion ( (step1 :primitive
                        (perform-mutual-indexing ?x ?y)))
        :conditions (
                (:use-when (indexed ?x) :at step1)
                (:use-when (indexed ?y) :at step1)
                (:use-when (isa xp ?x) :at step1)
                (:use-only-for-query
                        (explains ?explains-node ?x)
                 :at step1)
                (:use-only-for-query
                        (domain ?explains-node
                         ?explained-action)
                 :at step1)
                (:unsuperv (changed true ?explained-action
                 :at step1)
                        )
        :effects  (
                (step1 :assert (indexed-wrt ?x ?y)
                (step1 :assert (indexed-wrt ?y ?x
        :variables (?x ?y ?explains-node ?explained-action )
```

Figure 5: Mutual-indexing schemas

```
(actschema do-abstraction-change
        :todo   (abstracted ?r1 ?r2)
        :expansion ( (step1 :primitive (perform-abstraction ?r1 ?r2)))
        :conditions (
                (:use-when (isa relation ?r1) :at step1)
                (:use-when (isa relation ?r2) :at step1)
                (:use-when (relation ?r1 ?r1-type) :at step1)
                (:use-when (relation ?r2 ?r2-type) :at step1)
                (:use-only-for-query (domain ?r1 ?r1-domain) :at step1)
                (:use-only-for-query (co-domain ?r1 ?c) :at step1)
                (:use-only-for-query (co-domain ?r2 ?a) :at step1)
                (:use-only-for-query (parent-of ?c ?c-parent) :at step1)
                (:use-only-for-query (parent-of ?a ?a-parent) :at step1)
                (:use-when (equal ?r1-type ?r2-type) :at step1)
                (:use-when (equal ?c-parent ?a-parent) :at step1))
        :effects  (
                (step1 :assert (co-domain ?r1 ?c-parent))
                (step1 :assert (changed true ?r1-domain))
                (step1 :delete (co-domain ?r1 ?c))
                (step1 :delete (changed false ?r1-domain)))
        :variables (?r1 ?r2 ?r1-type ?r2-type ?r1-domain ?c ?a ?c-parent ?a-parent))
```

Figure 6: Abstraction schema

Therefore, the final learning plan Meta-AQUA constructs is (1) perform an abstraction transmutation on the new example of dog barking (realizing that dogs bark at containers); (2) perform EBG on the new explanation (producing a generalized version); (3) index the generalized XP in isolation; and finally, (4) use the new concept definition to mutually differentiate and index the two generalized explanations of why dogs bark. A subsequent story, such that a police officer and a canine enter a suspect's house, the dog barks at a garbage pail, and the suspect is arrested for possession of some marijuana found in the pail, causes no anomaly. Indeed, Meta-AQUA expects some type of contraband to be found in the container after it reads that the dog barked, but before it is told of its existence in the story. Thus, the learning improves both understanding and prediction.

5. This result supersedes the conjecture by Ram & Hunter (1992) that, unlike standard planning techniques, interactions and dependencies do not occur with learning goals.

4 Conclusions

Although Meta-AQUA is firmly in the CBR tradition, our approach diverges from it somewhat. Three elements traditionally characterize CBR. First, CBR usually processes instances or concrete episodic cases. However, some systems emphasize the integration of generalized knowledge and cases (e.g., Aamodt, 1993), and moreover, like Meta-AQUA, some CBR systems actually process abstract cases, including XPs (see Schank, Kass & Riesbeck, 1994). Secondly, CBR emphasizes the role of memory retrieval of past examples, rather than reasoning from first principles. This focus has led to research on indexing vocabulary and case adaptation. However, Meta-AQUA is a hybrid system that combines the CBR of the first two learning phases with the nonlinear planning of the third. Finally, traditional CBR systems stress goal-directed activity to focus both processing and learning (Kolodner, 1993; Ram & Hunter, 1992; Schank, 1982). Our approach to learning is also goal-directed, but in a very different style. Meta-AQUA is the first CBR system to specifically plan in the knowledge space using goals that specify changes in that space. Unlike INVESTIGATOR (Hunter, 1990), which creates plans in the external world to achieve learning goals (e.g., access a database to answer a question), Meta-AQUA's plans operate on the internal word of the system's background knowledge. Although many computational systems use a reflective reasoning approach (e.g., Collins, Birnbaum, Krulwich, & Freed, 1993; Fox & Leake, 1994; Oehlmann, Edwards, & Sleeman, 1994; Plaza & Arcos, 1993; Stroulia & Goel, in press), and a few have used the planning metaphor in learning (Hunter, 1990; Quilici, in press; Ram & Hunter, 1992; Ram & Leake, in press; Redmond, 1992), none of these systems have applied the planning metaphor as strictly as Meta-AQUA; none execute a planner like Nonlin on its own knowledge.

A number of advantages accrue from the mediation of learning through satisfaction of learning goals. First, learning goals decouple the many-to-many relationship between failures and algorithm. Secondly, an opportunistic approach to solving learning problems can be achieved by suspending the goals and resuming their pursuit at a time when satisfaction is more likely. Thirdly, learning goals allow chaining, composition, and optimization of the means by which learning goals are achieved. Fourthly, because nonlinear plans allows parallelism, learning algorithms may be executed concurrently. Finally, the use of learning goals allows detection of dependency relationships so that goal violations can be avoided.

Future research is directed toward incorporating more learning strategies. One of the weak points of the current system is that it reasons during learning at a macro-level. Meta-AQUA recognizes the functional difference between generalization and specialization and therefore can choose an appropriate algorithm based on which algorithm is most appropriate. For example, it cannot currently select between competing algorithms that both perform generalization. Meta-AQUA does not reason at the micro-level, as do systems that address the selective-superiority problem[6] in inductive learning (see, for instance, Brodley, 1993; Provost & Buchanan, 1992; Schaffer, 1993), although the scope of learning problems solved by Meta-AQUA is greater than these other systems.

Another limitation of the Meta-AQUA implementation is that learning self-evaluation (step 3 of Figure 1) does not exist. Thus, Meta-AQUA cannot cross-validate or compare various successful algorithms, nor can it currently judge when learning fails, and another algorithm must be chosen. Just as it detects, explains, repairs and learns from reasoning failures, an interesting line of future research is to allow Meta-AQUA to reason about its own learning.

To perform multistrategy learning, a CBR system must consider a number of factors that are not significant in isolated learning systems. In particular, a system must be able to handle insufficient resources and knowledge and manage dependency relations between learning algorithms at run-time. Many alternative solutions and interactions may occur, even when reasoning about simple situations. Treating the learner as a planner is a principled way to confront these difficulties. Many of the techniques and results from the planning literature can be appropriated in case-based systems to provide a better level of robustness and coverage in situations where many types of failure may occur. The aim is to transform these failures into opportunities to learn and improve the system's overall performance.

Acknowledgments

This research is supported by AFOSR under contract #F49620-94-1-0092 and by the Georgia Institute of Technology.

6. Empirical results suggest that various inductive algorithms are better at classifying specific classes or particular distributions of data than others. Each algorithm is good at some but not all learning tasks. The selective superiority problem is to choose the most appropriate inductive algorithm, given a particular set of data (Brodley, 1993).

References

Aamodt, A. (1993). Explanation-driven retrieval, reuse and learning of cases. In M. M. Richter, S. Wess, K.-D. Althoff, & F. Maurer (Eds.), *Proc. of the First European Workshop on Case-Based Reasoning*, Vol. 2 (pp. 279-284). SEKI Tech. Rep. No. SR-93-12 (SFB 314). Univ. of Kaiserslautern, Germany.

Brodley, C. E. (1993). Addressing the selective superiority problem: Automatic algorithm / model class selection. *Machine Learning: Proc. of the Tenth International Conference* (pp. 17-24). San Mateo, CA: Morgan Kaufmann.

Collins, G., Birnbaum, L. Krulwich, B. & Freed, M. (1993). The role of self-models in learning to plan. In A. L. Meyrowitz & S. Chipman (Eds.), *Foundations of knowledge acquisition* (pp. 117-143). Boston: Kluwer Academic Publishers.

Cox, M. T. (1993). *Introspective multistrategy learning* (CogSci. Tech. Rep. No. 2). Atlanta: Georgia Tech, College of Computing.

Cox, M. T., & Ram, A. (1992). Multistrategy learning with introspective meta-explanations. D. Sleeman & P. Edwards (Eds.), *Machine Learning: Proceedings of the Ninth International Conf.* (pp. 123-128). San Mateo, CA: Morgan Kaufmann.

DeJong, G., & Mooney, R. (1986). Explanation-based learning: An alternative view, *Machine Learning*, *1*(2), 145-176.

Fox, S., and Leake, D. (1994). Using Introspective Reasoning to Guide Index Refinement in Case-Based Reasoning. In *Proceedings of the Sixteenth Annual Conferences of the Cognitive Science Society* (pp. 324-329). Hillsdale, NJ: Lawrence Erlbaum.

Ghosh, S., Hendler, J., Kambhampati, S., and Kettler B. (1992). *UM Nonlin* [a Common Lisp implementation of A. Tate's Nonlin planner]. Maintained at the Dept. of Computer Science, University of Maryland, College Park, MD. Available by anonymous ftp from cs.umd.edu in directory /pub/nonlin.

Hammond, K. J. (1990). Learning and enforcement: Stabilizing environments to facilitate activity. B. W. Porter & R. Mooney (Eds.), *Machine Learning: Proc. of the Seventh International Conf.* (pp. 204-210). San Mateo, CA: Morgan Kaufmann

Hunter, L. E. (1990). Planning to learn. *Proc. of Twelfth Annual Conference of the Cognitive Science Society* (pp. 261-276). Hillsdale, NJ: Lawrence Erlbaum Associates.

Kolodner, J. L. (1993). *Case-based reasoning*. San Mateo, CA: Morgan Kaufmann Publishers.

Lebowitz, M. (1987). Experiments with incremental concept formation: UMIMEM. *Machine Learning*, 2, 103-138

Michalski, R. S. (1994). Inferential theory of learning: Developing foundations for multistrategy learning. In R. S. Michalski & G. Tecuci (Eds.), *Machine learning: A multistrategy approach IV* (pp. 3-61). San Francisco: Morgan Kaufmann Publishers

Mitchell, T., Keller, R., & Kedar-Cabelli, S. (1986). Explanation-based generalization: A unifying view, *Machine Learning*, *1*(1), pp. 47-80.

Oehlmann, R., Edwards, P., and Sleeman, D. (1994). Changing the viewpoint: Re-indexing by introspective questioning. In *Proc. of the 16th Annual Conf. of the Cognitive Science Society* (pp. 675-680). Hillsdale, NJ: Lawrence Erlbaum Associates

Plaza, E., & Arcos, J. L. (1993). Reflection and analogy in memory-based learning. In R. S. Michalski & G. Tecuci (Eds.), *Proc. of the Second Intl. Workshop on Multistrategy Learning* (pp. 42-49). Center for Art. Intel., George Mason Univ., Fairfax, VA

Provost, F. J., & Buchanan, B. G. (1992). Inductive policy. *Proc. of the Tenth National Conference on Artificial Intelligence* (pp. 255-261). Menlo Park, CA: AAAI Press.

Quilici, A. (in press). Toward automatic acquisition of an advisory system's knowledge base. *Applied Intelligence*.

Ram, A. (1991). A theory of questions and question asking. *The Journal of the Learning Sciences*, *1*(3&4), 273-318

Ram, A. (1993). Indexing, elaboration and refinement: Incremental learning of explanatory cases. *Machine Learning*, *10*(3), 201-248.

Ram, A., & Cox, M. T. (1994). Introspective reasoning using meta-explanations for multistrategy learning. In R. S. Michalski & G. Tecuci (Eds.), *Machine learning: A multistrategy approach IV* (pp. 349-377). San Francisco: Morgan Kaufmann.

Ram, A., & Hunter, L. (1992). The use of explicit goals for knowledge to guide inference and learning. *Applied Intelligence*, 2(1), 47-73.

Ram, A., & Leake, D. (in press). Learning, goals, and learning goals. In A. Ram & D. Leake (Eds.), *Goal-driven learning*. Cambridge, MA: MIT Press/Bradford Books.

Redmond, M. A. (1992). *Learning by observing and understanding expert problem solving* (Tech. Rep. GIT-CC-92/43). Doctoral dissertation. Atlanta: Georgia Institute of Technology, College of Computing.

Schaffer, C. (1993). Selecting a classification method by cross-validation. *Machine Learning*, *13*(1), 135-143.

Schank, R. C. (1982). *Dynamic memory: A theory of reminding and learning in computers and people*. Cambridge, UK: Cambridge University Press.

Schank, R. C., Kass, A., & Riesbeck, C. K. (1994). *Inside case-based explanation*. Hillsdale, NJ: Lawrence Erlbaum Associates

Stroulia, E., & Goel, A. (in press). Functional representation and reasoning for reflective systems. In *Applied Art. Intelligence*

Sussman, G. J. (1975). *A computer model of skill acquisition*. New York: American Elsevier.

Tate, A. (1976). *Project planning using a hierarchic non-linear planner*. Tech. Rep. #25, Dept. of Art. Intel., Univ. of Edinburgh

Published in the AAAI-94 Workshop on Case-Based Reasoning

A Comparative Utility Analysis of Case-Based Reasoning and Control-Rule Learning Systems

Anthony G. Francis, Jr. and **Ashwin Ram**
College of Computing, Georgia Institute of Technology
Atlanta, Georgia 30332-0280
{centaur, ashwin}@cc.gatech.edu

Abstract

The utility problem in learning systems occurs when knowledge learned in an attempt to improve a system's performance degrades performance instead. We present a methodology for the analysis of the utility problem which uses computational models of problem solving systems to isolate the root causes of a utility problem, to detect the threshold conditions under which the problem will arise, and to design strategies to eliminate it. We present models of case-based reasoning and control-rule learning systems and compare them with respect to the swamping utility problem. Our analysis suggests that CBR systems are more resistant to the utility problem than CRL systems.[1]

1. Introduction

All intelligent systems that learn can suffer from the utility problem, which occurs when knowledge learned in an attempt to improve a system's performance degrades performance instead (MINTON 1990). In this paper, we analyze the utility problem and examine its effects in case-based reasoners and control-rule problem solvers. Our methodology for this analysis couples a functional analysis of an AI system with a performance analysis of the system's algorithmic and implementational components. Such a computational model allows the identification of the root causes of the utility problem, which are combinations of algorithmic characteristics of an AI system (e.g., serial search of memory) with particular "parameters" that affect its operation (e.g., knowledge base size). Identifying the precise algorithmic nature of a root cause allows us to predict the threshold conditions under which it will affect a system severely enough to cause a utility problem.

Our analysis provides a general and theoretical framework for addressing this problem in systems that have been studied empirically (e.g., control-rule learning (CRL) systems) and in systems for which little utility analysis has been performed (e.g., case-based reasoning (CBR) systems). We use this framework to compare CBR and CRL systems, and find that CBR systems are more resistant to the utility problem than CRL systems.

2. Analyzing the Utility Problem

The utility problem was first detected in PRODIGY/EBL (MINTON 1988). PRODIGY/EBL is a *control-rule learner*,[2] a type of system that attempts to improve its problem-solving performance by learning search-control knowledge, called *control rules*, that reduce the amount of search it needs to perform by eliminating dead-end paths and selecting profitable ones. What Minton and others noticed about systems like PRODIGY/EBL was that the system could actually get slower after having learned control rules, rather than faster. At each step in the search space, a control-rule problem solver has to match all of its control rules against the current state to determine if they should fire. As that library of control rules grows in size, the cost of matching the control rules increases to the point that it outweighs or "swamps" the savings in search they provide.

This side effect of learning was called the "utility problem": learning designed to improve the system's performance ended up degrading performance instead (HOLDER ET AL. 1990). Since Minton's discovery of this "swamping" utility problem in PRODIGY, researchers have identified many different types of utility problems, each manifesting itself in slightly different ways. Because some types of utility problems are affected by the hardware architecture of the system and others are largely independent of hardware concerns, we can group the different types of utility problems into two rough classes: *architectural utility problems*, which arise from interactions between a system's learning and its hardware architecture, and *search-space utility problems*, which arise from interactions between learning and problem solving algorithms. A full discussion of the different types of utility problems is contained in (FRANCIS & RAM 1994). In this paper, we will focus on swamping, which is the utility problem most commonly encountered in learning systems. We will reserve the term "the utility problem" for the general utility problem, and will refer to specific versions of the utility problem, such as swamping, by their names.

3. A Methodology for Utility Analysis

We propose the use of algorithmic complexity theory as a tool for the analysis of the general utility problem. Our methodology involves analyzing different types of AI systems and decomposing their cognitive architectures into lower-level functional units, including problem-solving engines and memory systems, that can be repre-

[1]This research was supported by the United States Air Force Laboratory Graduate Fellowship Program and the Georgia Institute of Technology. A longer version of this paper (FRANCIS & RAM 1994) is ftp-able from ftp.cc.gatech.edu:/pub/ai/ram/git-cc-94-14 ps Z

[2] Etzioni (1992) uses the term meta-level problem solvers for control-rule learning systems. We have avoided this term because of the possible confusion with metacognition, which includes systems that "know what they know" (metaknowledge) and systems that reason about their own reasoning processes (metareasoning, or introspection).

sented by formal algorithmic models. Our algorithmic approach incorporates both functional-level aspects of the computation, such as the system's cognitive architecture and its knowledge base, and implementation-level aspects, such as the performance characteristics of the system's hardware architecture. This multi-level analysis is crucial for the study of the utility problem because many utility problems arise due to interactions between the functional level of the system and the way that functional computation is actually implemented. Our methodology can be used to identify potential utility problems as well as to design coping strategies to eliminate their effects. In this paper, we focus on a comparative analysis of case-based reasoning and control-rule systems.

3.1. AI Systems and Learning

Formally, we can describe an AI system as a triple (**CA**, **KB**, **HA**). The *cognitive architecture* **CA** specifies a system in terms of separate functional modules that carry out fixed subtasks in the system, while the *knowledge base* **KB** represents the internal "data" that the **CA** uses to perform its computations. The *hardware architecture* **HA** defines the operations that a system can perform at the implementation level, as well as their relative costs. The cost (and hence the utility) of an operation may be different on different **HA**'s: for example, retrieval might take longer on a serial machine than it would on a parallel machine.

3.2. Utility and the Utility Problem

Utility can only be defined in terms of "performance" measures that judge the efficiency of a reasoner, such as execution time, number of states searched, storage space used, or even quality of solution. These *evaluation metrics* measure the costs that a system incurs during its reasoning. Given a particular evaluation metric, the *utility* of a learned item can be defined as the change in expectation values of a problem solver's performance on the metric across a problem set (MARKOVITCH & SCOTT 1993). To compute the utility of a change to the system's knowledge base with respect to some metric, we want to compute the costs that the system will incur for different problems weighted by the probability that the system will actually encounter those problems. Thus, utility is a function not only of the learned item but also of the learning system, the problem set, problem distribution, and the evaluation metric. The *utility problem* occurs when a learning system makes a change to its **KB** with the goal of improving problem solving utility on some metric by a calculated improvement F_c, but which has the side effect of degrading problem solving utility for another (possibly identical) metric by some actual amount F_a that outweighs the savings (i.e., $F_c<F_a$).

3.3. Dissecting the Utility Problem

In general, utility problems are not global, emergent properties of computation but can instead be tied to specific interactions between the **CA**, the **KB**, and the performance characteristics of the **HA**. In a CRL system, the interaction of interest is the relationship between match time and knowledge base size; in a CBR system, a similar interaction exists between case retrieval time and case library size.

We can formally define an *interaction* to be a combination of a set of parameters, a module, and a set of effects. The *module* represents the part of the **CA** that is responsible for the relationship between the parameters and their effects. *Parameters* represent characteristics of the system's knowledge base, while *effects* represent the performance measures that affected by the interaction. Thus, an interaction defines a function between learning (changes in the knowledge base) and performance (changes in the evaluation metric), mediated by the characteristics of the algorithmic component of the interaction (the module).

Utility problems arise when a learning module in the system causes parameter changes which interact with some **CA** component to produce "side" effects that impact the performance measures a learning module is designed to improve. This kind of coupling between a learning module and an interaction is a potential *root cause* of a utility problem. For a particular root cause, the calculated improvement F_c is the savings that the learning module is designed to perform, while the actual cost F_a is the actual change in performance taking into account the side effects of the interaction.

By comparing the algorithmic behavior of the learning module, the root cause interaction it is paired with, and the cost and savings functions that they contribute, we can compute *threshold conditions* — limiting values for the parameter changes that the system can tolerate before the actual costs exceed the calculated improvement and the system encounters a utility problem. Eliminating the general utility problem involves identifying the root causes of particular utility problems that can arise in a system and designing *coping strategies* that prevent their threshold conditions from being satisfied.

4. Modeling CRL and CBR Systems

To compare CBR and CRL systems, we must develop computational models of these systems and compare learning and problem solving in each.

The baseline for comparison of the computational model approach is the *unguided problem solver*. Unguided problem solvers (UgPS) are a class of problem solvers that operate without search control knowledge. The algorithm of a UgPS is a knowledge-free weak

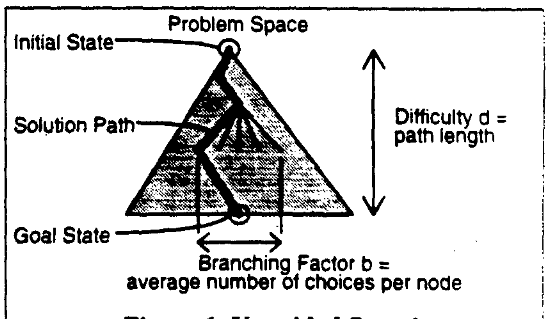
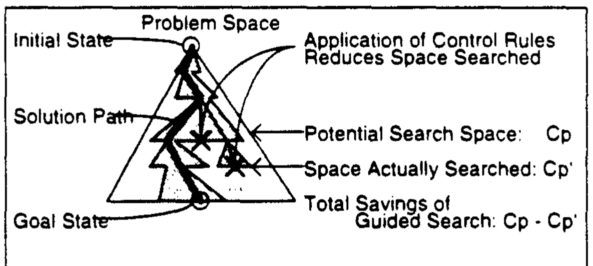

Problem Space
Initial State
Solution Path
Goal State
Difficulty d = path length
Branching Factor b = average number of choices per node

Figure 1. Unguided Search

method guaranteed to find a solution if one exists, such as breadth-first search; the only knowledge it uses is its *operator library*. For a given problem p with a solution of path length d operating in a search space with branching factor b, a UgPS will examine b^d nodes during search (see Figure 1). The number of nodes that the system expands is termed the *complexity* of a problem and is denoted C_p. Because the UgPS solves problems in exponential time, it can serve as a "baseline" against which more efficient learning systems can be compared. Much of "intelligence" can be viewed as attempts to reduce this combinatorial explosion through the use of heuristics or other techniques (NEWELL & SIMON 1975; RAM & HUNTER 1992; SCHANK & ABELSON 1977).

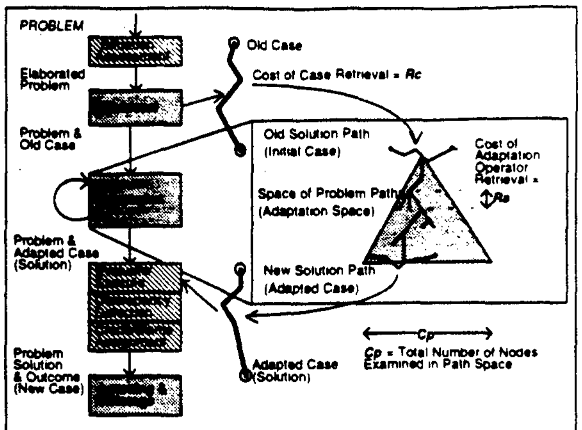
4.1. Control-Rule Problem Solvers

Learning systems improve over the UgPS by finding ways to reduce or eliminate search. A CRL system reduces search by retrieving and applying *control rules* at each state it visits during problem solving, giving it the ability to select or reject states. This control knowledge is a completely different kind of knowledge than operator knowledge and must be stored in a separate *control rule library*. If a system's control rule library is empty and control rules are not available, the problem solver resorts to blind search. Once a solution path has been found, correct decisions can be cached in the library as control rules that will guide the system in the future (see Figure 2). This model, while simplified, approximates many existing systems, including Soar and Prodigy.


Problem Space
Initial State
Application of Control Rules Reduces Space Searched
Solution Path
Potential Search Space: Cp
Space Actually Searched: Cp'
Total Savings of Guided Search: Cp - Cp'
Goal State
Figure 2. Search Guided by Control Rules

4.2. Case-Based Reasoners

Case-based reasoning is primarily experience-based; when a CBR system encounters a new problem, it checks its *case library* of past problem solving episodes, or *cases*, looking for a similar case that it can adapt to meet the needs of the new problem. Our model of CBR has two primary knowledge libraries: the case library itself, indexed so that the most appropriate case can be retrieved in new problem-solving situations, and an *adaptation library* that stores *adaptation operators* that are used to transform the cases once they are retrieved. When a CBR system is presented with a problem, it retrieves an appropriate past case based on the problem's features, its goals, and the indices it has in its case library. Then, the case is adapted by performing search in the space of problem paths: the adaptation operators transform entire paths into new paths until a satisfactory solution path is achieved (see Figure 3). Once the new solution is found, it is stored in the case library indexed for future retrieval.


PROBLEM
Elaborated Problem
Problem & Old Case
Problem & Adapted Case (Solution)
Problem Solution & Outcome (New Case)
Old Case
Cost of Case Retrieval = Rc
Old Solution Path (Initial Case)
Space of Problem Paths (Adaptation Space)
New Solution Path (Adapted Case)
Cost of Adaptation Operator Retrieval = Ra
Adapted Case (Solution)
Cp
Cp = Total Number of Nodes Examined in Path Space
Figure 3. Search in the Space of Problem Paths

5. Analyzing Retrieval Costs

Our analysis of these models focuses on retrieval costs in CRL and CBR systems—how many retrievals are made, and how much does each retrieval cost? Retrieval is often cited as the core source of power for CBR systems, yet the retrieval cost is a critical factor in the swamping utility problem. An analysis of retrieval costs in CRL and CBR systems before and after learning reveals interesting differences in how each deals with retrieval.

For our analysis, we will assume that both the CRL and CBR systems operate on the same problem set, and that their problem spaces are defined by the same operator library. We further assume that the costs of adaptation operators are roughly equivalent to those of regular operators; this assumption may or may not be true for particular systems but is reasonable for this analysis.

We define a basic operation of retrieval, R, which chooses an item in a knowledge library based on some matching function. In general, for a given hardware architecture **HA**, the cost of retrieval for a knowledge library i, denoted R_i, is a function of both the library i and the item to be retrieved, r: $R_i = f(r,i)$. For a serial hardware architecture, $\mathbf{HA_S}$, the most important variable in this cost function is the number of items in the knowledge library, K_i. We will approximate this serial cost function with $R_i = cK_i$, where c is a constant multiplier that approximates the (nearly) linear cost function for matching on serial systems like $\mathbf{Ha_S}$.[3]

The particular interaction we will examine is the *retrieval time interaction*: the relationship between the retrieval operation R_i, knowledge library size K_i, and running time t. Because the learning operations in CBR and CRL systems have the effect of increasing the size of knowledge libraries in the system, their learning modules coupled with the retrieval time interaction form *po-*

[3]An indexed or parallel memory system might improve on this retrieval function (or, then again, might not, depending on the domain; see DOORENBOS 1993 and TAMBE ET AL. 1990)

tential root causes of the utility problem. Now, let's examine the dynamics of learning and retrieval in CBR and CRL systems and attempt to establish the *threshold conditions* for the utility problem in each.

5.1. Retrieval in Control-Rule Problem Solvers

In its initial state, without control rules, a CRL system is equivalent to the UgPS. It searches C_p states, retrieving a set of operators at each step with a cost of R_o. No control rules exist, so the cost of control-rule retrieval $R_c = 0$. Thus, the total cost in retrievals of the initial system is C_pR_o. After the system has learned a set of control rules, it has the capacity to guide its search. The number of states searched is reduced to C_p', where $C_p' < C_p$. However, in addition to retrieving a set of operators, it also needs to retrieve control rules at each step; thus, the cost for solving a problem rises to $C_p'(R_o+R_c')$.

The expected savings that guided problem solving brings are the costs of the states that the problem solver avoids: $(C_p - C_p')R_o$. The added costs are the costs of matching the control rules at each step: $C_p'(R_c' - R_c) = C_p'\Delta R_c$. (Note: because $R_c = 0$, $\Delta R_c = R_c'$). The utility problem will arise when the added costs exceed the expected savings. Thus, the threshold condition is $(C_p - C_p')R_o < C_p'\Delta R_c$; in other words, the swamping utility problem arises when the added cost of retrieval outweighs the benefits of individual rules. But will this threshold condition ever be met?

In the limit, the maximum search reduction is to a single path ($C_p' = d$), and operator retrieval costs are constant ($R_o' = R_o$) since the library of operators the system uses does not change in size. The *maximum expected savings* possible for any problem are thus $(C_p - d)R_o$. In contrast, the cost of retrieving control rules (R_c') increases without bound as the control base K_c increases in size; in the limit, the added costs associated with a rulebase are $dR_c' = dc(K_c)$ and thus can outweigh the maximum possible savings. Therefore, the threshold conditions *can* be met and the CRL system will encounter the utility problem.

These results indicate that swamping is a function of the potential speedup of learned items, the cost function of retrieval (which is itself dependent on retrieval strategies and machine architecture), and the number of items a system needs to learn. If the system converges on a bounded set of learned items and the hardware slowdown never approaches the utility of those items, the system will never be swamped.[4] If the learned items are of low utility, or if the learner never converges on a bounded set, as might be the case for an open-world or multidomain system, then the swamping problem can eliminate the benefits of the learned rules.

[4] For example, on a parallel machine with a logarithmic cost function $R_i = c(\log K_i)$, the threshold condition $(C_p - C_p')R_o < C_p'c(\log K_c)$ may never be met in a closed-world domain in which a small set of knowledge items learned by rote are adequate for performance. If the learning system successfully converges on a small enough set, the logarithmic slowdown will be negligible compared to the potential savings. This condition can arise on serial architectures as well, but because the cost function is linear in the size of the knowledge base the constraints on the size of the learned set are much more severe.

5.2. Retrieval in Case-Based Reasoners

To analyze utility effects in CBR systems, we need to measure the performance of a CBR system as it learns. To provide a basis for this measurement, we assume that a CBR system that does not have an appropriate case in memory can resort to some method (e.g., adaptation of a "null case" or "from-scratch" reasoning) that is cost-equivalent to the UgPS. Many existing CBR systems have such a last-resort method (e.g., KOTON 1989, KOLODNER & SIMPSON 1988).

A CBR system that resorts to null-case adaptation beginning with no experiences must still incur the cost of retrieving the null case (R_c) and then search the space of problem paths until the case has been adapted into a satisfactory solution. Under our earlier assumptions, the total number of paths the system examines is C_p, and one adaptation retrieval (R_a) occurs per step. Thus, the total cost of case adaptation before learning is $R_c + C_pR_a$. After the system has learned a library of cases, it will still need to retrieve a case but each case will require much less adaptation, reducing the number of paths examined to C_p' where $C_p' << C_p$. The cost of retrieving cases will increase to R_c' where $R_c' > R_c$. Thus, the total costs are $R_c' + C_p'R_a$.

To evaluate these results we must again examine the benefits and costs of case retrieval. The expected savings are the costs of the states that the problem solver avoids: $(C_p - C_p')\ R_a$, while the added costs are the increased costs of retrieval of cases $R_c' - R_c = \Delta R_c$. In the limit, the cost of retrieval increases without bound as the case library increases in size: $R_c' = c(K_c)$. However, as we approach the limit the case library contains many appropriate cases and little adaptation needs to be done—perhaps only one or two steps. In general, whenever the threshold condition $(C_p - C_p')\ R_a < \Delta R_c$ is met, the cost of retrieval outweighs the benefits of case adaptation; under these conditions, CBR systems will be swamped.

5.3. Advantages of Case-Based Reasoning

While this analysis reveals that both control-rule learners and CBR systems can suffer from the swamping utility problem, it also reveals that CBR systems have important advantages over CRL systems.

The primary advantage CBR systems have over control-rule problem solvers is that cases are retrieved only once during the lifetime of problem solving. For a control rule problem solver to avoid swamping, the increase in cost of retrieval of a control rule must be less than the fraction of total states that the system avoids in guided problem solving times the cost of an operator: $\Delta R_c < R_o(C_p - C_p')/C_p'$. For a CBR system, on the other hand, the increase in cost of a case retrieval must be less than the cost of the number of adaptation steps avoided: $\Delta R_c < R_a(C_p - C_p')$. The missing C_p' term in the denominator of the CBR equation arises because the increased cost of retrieval of control rules are incurred at each step in the search space, whereas the increased cost of case retrieval is incurred only once during problem solving for a CBR system. In other words, CBR systems *amortize* the cost of case retrieval across all adaptations, making them much more resistant to increases in retrieval costs than CRL systems.

This amortization also makes CBR more amenable to solutions to the swamping problem, such as deletion policies or indexing schemes. In order to be effective, any coping strategy needs to reduce retrieval time to the point that the threshold conditions are never satisfied. For CRL systems, this upper limit is $R_c' < R_o(C_p - C_p')/C_p'$; for CBR systems, this upper limit is $R_c' < R_a(C_p - C_p')$, a much higher (and hence much less stringent) limit on the maximum time retrieval can take for a system to be guaranteed to avoid swamping.

5.4. Future Comparative Analysis

The above analysis does not close the book on the utility problem in CBR and CRL systems. There are a number of other differences that can contribute to the utility problem, including differences in operator costs, degree of search reduction, and knowledge base size.

Normal problem solving operators and adaptation operators do not necessarily have comparable costs. A traditional problem solving operator makes a change to a single state, while an adaptation operator can potentially make several changes to a case. Thus, the same reduction in C_p' in CBR and CRL systems could provide different degrees of savings because the individual steps being saved may not cost the same.

Another difference between CBR and CRL systems is that they are not guaranteed to produce the same reduction in C_p' after being exposed to the same set of training examples. While both types of systems are senstitive to the structure of the domain, search reduction in CRL systems depends on the learning and matching policy for control rules, but in CBR systems depends on the indexing scheme and similarity metric used for case retrieval. Therefore, the same training set might produce different search reductions for CRL and CBR systems.

Furthermore, depending on the learning biases chosen, different systems could extract varying numbers of cases or rules from the same set of examples. Therefore, even if the search reduction is the same, there is no guarantee that the size of the systems' knowledge libraries will be comparable, and hence no guarantee that their performance on the utility problem will be identical.

Providing a principled account of the effect contributing factors like these have on the utility problem is the primary goal of our research. While we believe we have identified one factor — amortization — which contributes to the performance differences of CBR and CRL systems, it is only one contributing factor among many that determine the systems' respective performances. Accounting for other contributing factors, such as operator costs, degree of search reduction, and knowledge base size, will allow a principled comparison of CBR and CRL systems performance on the utility problem.

For example, there have been few attempts to systematically evaluate the cost-utility tradeoffs in CBR systems with very large case libraries. However, reports of the speedup provided by cases (e.g., KOTON 1989) and control rules (e.g., TAMBE ET AL 1990) suggest that cases can provide large improvements, up to an order of magnitude greater than the speedups provided by control rules. It is *possible* that the utility of cases may be high enough to allow CBR to avoid swamping, but it is not clear whether this will always be the case.

6. The Bottom Line

CBR's patterns of retrieval make it resistant to the utility problem. Because the cost of case retrieval is amortized over many adaptation steps, ideal CBR systems suffer less severely from the same overhead and are more amenable to coping strategies than CRL systems. This analysis suggests several future lines of comparative research on CBR and CRL, such as operator costs, degree of search reduction and size of knowledge bases.

References

Doorenbos, R.B. (1993) Matching 100,000 Learned Rules. In *Proceedings of the Eleventh National Conference on Artificial Intelligence*, p290-296. AAAI Press/MIT Press.

Etzioni, O. (1992) An Asymptotic Analysis of Speedup Learning. In *Machine Learning: Proceedings of the 9th International Workshop*, 1992.

Francis, A. and Ram, A. (1994). Computational Models of the Utility Problem and their Application to a Utility Analysis of Case-Based Reasoning. *Technical Report GIT-CC-94-24*. College of Computing, Georgia Institute of Technology

Holder, L.B.; Porter, B.W.; Mooney, R.J. (1990) The general utility problem in machine learning. In *Machine Learning Proceedings of the Seventh International Conference*, 1990

Kolodner, J.L. & Simpson, R.L. (1988). The Mediator A case study of a case-based reasoner. Georgia Institute of Technology, School of Information and Computer Science, Technical Report no. GIT-ICS-88/11. Atlanta, Georgia.

Kolodner, J.L. (1993). *Case-based Reasoning*. Morgan Kaufmann, 1993.

Koton, P. A. (1989). A Method for Improving the Efficiency of Model-Based Reasoning Systems. Laboratory for Computer Science, MIT, Cambridge, MA. Hemisphere Publishing

Markovitch, S. and Scott, P.D. (1993) Information Filtering Selection Methods in Learning Systems. *Machine Learning*, 10: 113-151.

Minton, S. (1988). Quantitative results concerning the utility of explanation-based learning. In *Proceedings of the Seventh National Conference on Artificial Intelligence*, 1988

Minton, S. (1990). Quantitative results concerning the utility of explanation-based learning. *Artificial Intelligence*, 42(2-3), March 1990.

Newell, A. & Simon, H.A. (1975). Computer Science as Empirical Inquiry: Symbols and Search. Reprinted in Haugeland, J., (ed.). *Mind Design: Philosophy, Psychology, Artificial Intelligence*, chapter 1, pp 35-66. MIT Press, 1981

Ram, A. & Hunter, L. (1992) The Use of Explicit Goals for Knowledge to Guide Inference and Learning. *Applied Intelligence*, 2(1):47-73.

Schank, R. & Abelson, R. (1977). *Scripts, Plans, Goals and Understanding*. LEA, Hillsdale, NJ.

Tambe, M.; Newell, A.; Rosenbloom, P. S. (1990) The Problem of Expensive Chunks and its Solution by Restricting Expressiveness. Machine Learning, 5:299-348, 1990

In M. T. Cox & M. Freed (Eds.), *Proceedings of the 1995 AAAI Spring Symposium on Representing Mental States and Mechanisms* (pp. 22-30). Menlo Park, CA: AAAI Press.

Representing Mental Events (or the Lack Thereof)

Michael T. Cox
Cognitive Science Program
Georgia Institute of Technology
Atlanta, Georgia 30332-0505

Abstract

This paper focusses upon the level of granularity at which representations for the mental world should be placed. That is, if one wishes to represent thinking about the self, about the states and processes of reasoning, at what level of detail should one attempt to declaratively capture the contents of thought? Some claim that a mere set of two mental primitives are sufficient to represent the utterances of humans concerning verbs of thought such as "I forgot her name." Alternatively, many in the artificial intelligence community have built systems that record elaborate traces of reasoning, keep track of knowledge dependencies or inference, or encode much metaknowledge concerning the structure of internal rules and defaults. The position here is that the overhead involved with a complete trace of mental behavior and knowledge structures is intractable and does not reflect a reasonable capacity as possessed by humans. Rather, a system should be able instead to capture enough details to represent a common set of reasoning failures. I represent a number of examples with such a level of granularity and describe what such representations offer an intelligent system. This capacity will enable a system to reason about itself so as to learn from its reasoning failures, changing its background knowledge to avoid repetition of the failure. Two primitives are not sufficient for this task.

Introduction

An early tenet of artificial intelligence is that reasoning about the world is facilitated by declarative knowledge structures that represent salient aspects of the world. An intelligent system can better understand and operate in such a represented world as opposed to one in which knowledge is encoded procedurally or implicitly. The system may inspect and manipulate such structures, the system can be more easily modified and maintained (by either its programmer or itself), and such representations provide computational uniformity.[1] Furthermore, if a system is to reason about itself, the above tenet can be applied to representations of its own reasoning and knowledge. The aim of this paper, therefore, is to begin to outline a declarative representation of mental activity. The goal is to explicitly represent the mental world that reasons about the physical world, just as past research has explicitly represented the physical world itself. Thus, instead of representing states and events in the physical world, this paper discusses how and at what grain level one should represent mental states and mental events.

A large number of terms exist in the English language that concern mental activity. Although I do not wish to equate surface features of a language utterance with process or states that may or may not lie behind a given utterance, a number of English expressions point to interesting problems for declarative representations. For this paper I wish to examine a few "cognitively pure" terms that generally refer only to the internal world of the reasoner, rather than the external world of physical objects and other people.[2] This paper will also ignore non-cognitive mental states such as emotions (affect, e.g, fear and love). Rather, it will focus on more simple concepts such as think, forget, and imagine, although humans are likely to think thoughts about the external world, forget to perform actions in the world, and imagine what the physical world may be like. With such constraints, the hope is to partially insulate the task by avoiding consideration of the more complex terms that intertwine the internal and external worlds, and instead, attempt to sketch an ontology of mental representations and a vocabulary of the content of such representations.

Many cognitive vocabularies make a prominent distinction between mental states (as knowledge or belief) and mental mechanisms (as the mental events that process knowledge or information). For example, Conceptual Dependency (CD) theory (Schank, 1975) distinguishes between two sets of representations: primitive mental ACTs and mental CONCEPTUALIZATIONs upon which the ACTs operate. In addition, the theory proposes a number of causal links that connect members of one set with members of the other. With such building blocks, a representational language such as CD must be able to represent many process

1. Proponents of procedural representations have argued against these points and countered with advantages of their own. See Winograd (1975/1985), especially his argument that second-order knowledge is easier to represent procedurally. See Stein & Barnden (this volume) for arguments in favor of the procedural representation of some knowledge via mental simulation or projection of hypothetical events.

2. Certainly the boundary between the two worlds is not a very clean line. Terms such as "speak" concern the manipulation of mental terms (e.g., concepts), but complicate the representation with details of expression, translation, interpretation and the physical means of conveyance.

terms: think(about), remember, infer, realize and calculate; and numerous state terms: fact, belief, guess, doubt, and disbelief. This paper will refer to the execution of any mental process (or arbitrarily long string of processes) by the generic term "cognize,"[3] and to a CONCEPTUALIZATION simply by the term "state." See Figure 1 for a preliminary sketch of a target ontological vocabulary for mental representations. If a reasoner is to understand its own reasoning and mental conditions in any substantial level of detail, it will require a semantic hierarchy containing representations for most of these cognitive terms.

In addition to the state-process dichotomy, process terms are often subdivided by function into mental events that involve memory and transfer of information and those that involve computation or inference. Inferential processes are usually associated with logical or hypothetical reasoning. Example terms include hypothesize, speculate, deduce, corroborate, and postulate. However, our desired vocabulary includes additional terms that receive little attention in the artificial intelligence community. In Figure 1, inferential processes are subdivided into those that are driven by a deliberate goal for processing (Calculate) and those in which belief is either more incidental or less exact (Realize).

Until recently, examples of memory processes such as remember, remind, recall, recognize, and forget (but here, the lack of an event) have been largely unexamined and without explicit representation. Especially in the context of case-based reasoning or any problem solving that depends on indexed memory hierarchies to support a performance task,

3. The general term "to think" is an unfortunately overloaded term. It can be used in the sense of "to think about" (and thus refers to a generic mental process) or "I think so." (refers to some qualitative level of confidence). Therefore, cognize is a less ambiguous representational label.

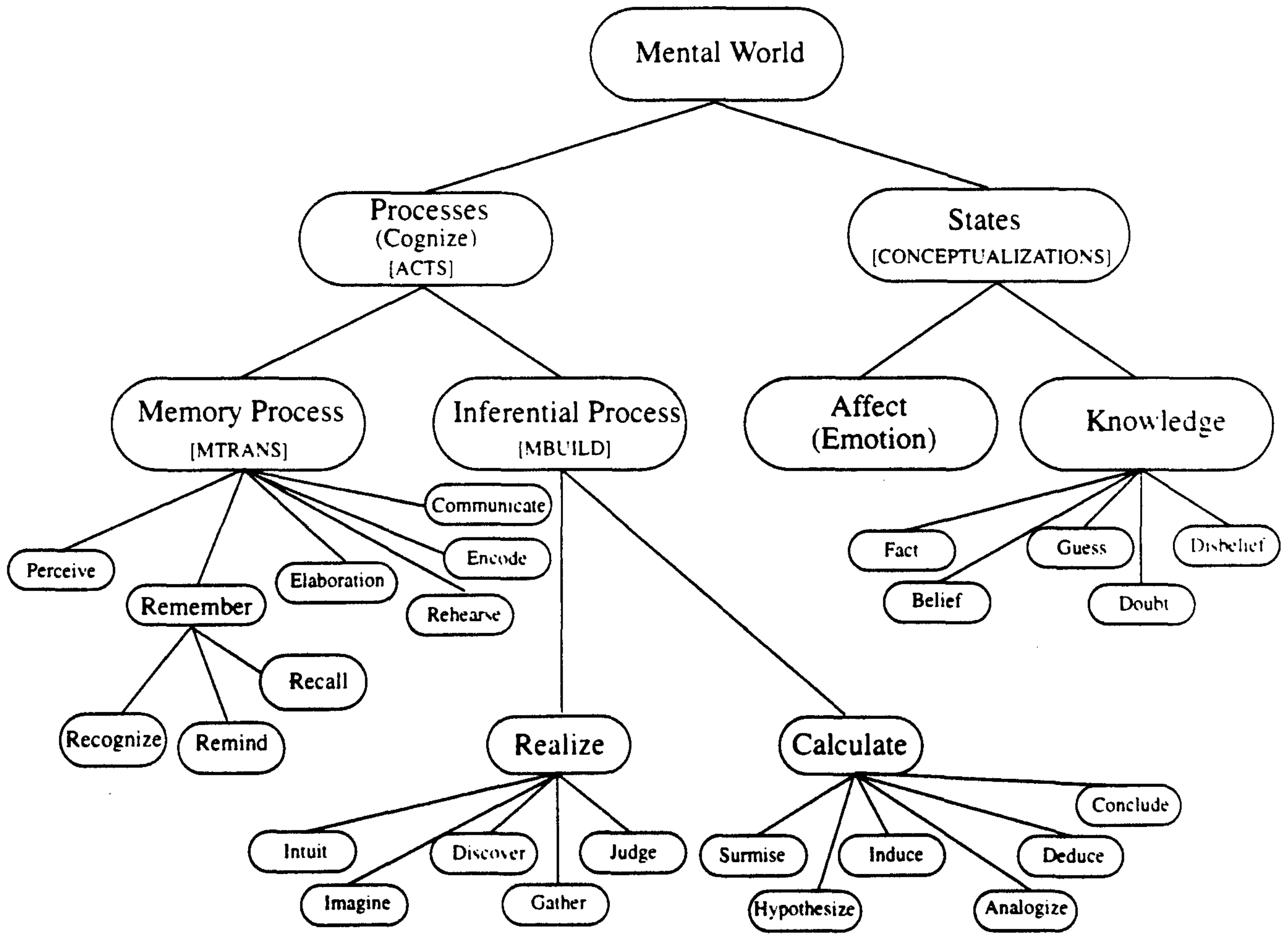


Figure 1. Preliminary partial ontology of mental terms

understanding the operation of memory can be of benefit when learning (see also Leake, this volume; Fox & Leake, this volume; and Kennedy, this volume, for additional arguments in favor of this position). A system that is to adjust the organization of memory will have a better chance of success if it has knowledge of the function and operation of the (cognitive) device it is changing. Therefore, for a system to understand and change its own memory effectively, it is important that the system to be able to represent the memory processes explicitly.

Representing Forgetting: An example

In order to use representations of mental terms effectively, a system should consider the structure of the representation, rather than simply how a system can syntactically manipulate representations or make sound inferences from them. For instance, it is not very useful to simply possess a predicate such as "forget" or "remember" when trying to understand memory failure.

$$\text{Forget (John, M)}$$
$$\neg\ \text{Remember (John, M}$$

Because the predicates involve memory, it may be helpful to posit the existence of two contrasting sets of axioms: the background knowledge (BK) of the agent, P, and the foreground knowledge (FK) representing the currently conscious or active axioms of the agent. Then, the interpretation of forget becomes

$$\text{Forget (P, M)} \rightarrow \exists M.\ (M \in BK_p) \wedge (M \notin FK_p)$$

But to add this interpretation is to add content to the representation, rather than simply semantics. It is part of the *metaphysical interpretation* (McCarthy & Hayes, 1969) of the representation that determines an ontological category (i.e., what ought to be represented), and it begins to provide epistemological commitments (e.g., that the sets BK and FK are necessary representational distinctions). However, meaning is not only correspondences with the world to be represented, but meaning is also determined by the inferences a system can draw from a representation (Schank, 1975). The forget predicate does little in this regard. Moreover, these predicates will not assist a reasoning system to understand what happens when it forgets some memory item, M, nor will they help the system learn to avoid forgetting the item in the future. Finally, the representation of a mental event which did not actually occur is not well captured by a simple negation of a predicate representing an event which did occur (Cox & Ram, 1992b), thus $\neg$ Remember (John, M) is essentially a vacuous proposition. This is not to say that logic cannot represent such a mental "non-event," rather, I simply wish to illustrate that it is not an elementary task to construct an adequate representation for forgetting and that a single logical predicate will not suffice.

An alternative approach was undertaken by Schank, Goldman, Rieger & Riesbeck (1972) in order to specify the representations for all verbs of thought in support of natural language understanding. They wish to represent what people say about the mental world, rather than represent all facets of a complex memory and reasoning model. They therefore use only two mental ACTS, MTRANS (mental transfer of information from one location to another) and MBUILD (mental building of conceptualizations), and a few support structures such as MLOC (mental locations, e.g., Working Memory, Central Processor and Long Term Memory).[4]

As a consequence, the representation by Schank et al. of forgetting is as depicted in Figure 2. John does not mentally transfer a copy of the mental object, M, from the recipient case of John's Long Term Memory to his Central Processor. Such a representation does provide more structure than the predicate forms above, and it supports inference (e.g., if M was an intention to do some action, as opposed to a proposition, then the result of such an act was not obtained: Schank, 1975, p. 60), but the CD formalism cannot distinguish between the case during which John forgot due to M not being in his long-term memory and a case of forgetting due to missing associations between cues in the environment and the indexes with which M was encoded in memory. It does not provide enough information to learn from the experience.

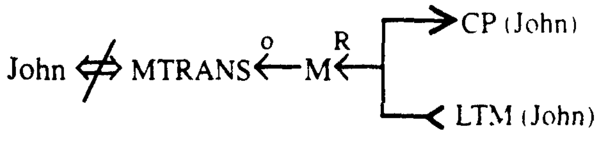


Figure 2. CD representation of forgetting
o=mental object or conceptualization. R=Recipient. CP=Central Processor; LTM=Long Term Memory

Cox (1994) and Cox and Ram (1992b) pose an alternative representation for such mental phenomena using Explanation Pattern (XP) theory (Leake, 1992; Ram, 1991; Schank, 1986; Schank, Kass, & Riesbeck, 1994). The Meta-XP structure of Figure 2[5] represents a memory retrieval attempt enabled by goal, G, and cues, C, that tried to retrieve some memory object, M, given an index, I, that did not result in an expectation (or interpretation), E, that should have been equal to some actual item, A. The fact that E is out of the set of beliefs with respect to the reasoner's foreground knowl-

4. Schank et al. (1972) actually referred to Working Memory as Immediate Memory and the Central Processor as a Conceptual Processor. I have used some license to keep terms in a contemporary language. Moreover, Schank et al. used a third primitive ACT, CONC, which was to conceptualize or think about without building a new conceptualization, but Schank (1975) dropped it. For the purposes of this paper, however, the differences do not matter.

edge (FK), that is, is not present in working memory, initiates the knowledge that a retrieval failure had occurred.

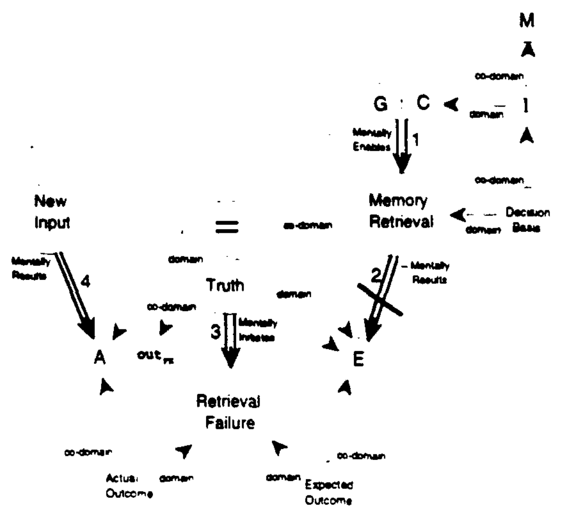


Figure 3. Meta-XP representation of forgetting
A=actual; E=expected; G=goal; C=cues; M=memory item, I=memory index.

This representation captures an entire class of memory failures: failure due to a missing index, I; failure due to a missing object, M; failure because of a missing retrieval goal, G;[6] or failure due to not attending to the proper cues, C, in the environment. Such a representation allows the system to reason about these various causes of forgetting; it can inspect the structural representation for a memory failure and therefore analyze the reasons for the memory failure. Such an ability facilitates learning because it allows a learner to explain the reasoning failure and use the result in determining what needs to be learned to avoid the failure in the future (Ram & Cox, 1994).

5. Meta-XPs are essentially directed graphs with nodes being either states or processes and enables links connecting states with the processes for which they are preconditions, results links connecting a process with a result, and initiate links connecting two states. Numbers on the links indicate relative temporal sequence. Attributes and relations are represented explicitly in these graphs. Thus, the ACTOR attribute of an event X with some value Y is equivalent to the relation ACTOR having domain X and co-domain Y. See Cox & Ram (1992a; 1992b) and Ram & Cox (1994) for further representational details.
6. The agent never attempted to remember. For instance, the reasoner may have wanted to ask a question after a lecture was complete, but failed to do so because he never generated a goal to remember once the lecture was complete. Alternatively the agent may know at the end of the lecture that he needs to ask something, but cannot remember what it was. This second example is the case of a missing index. Note that both a missing index and an incorrect index may be present at the same time. In such cases, a target item is not retrieved, whereas an interfering item is retrieved instead.

However, complete representations for all inferences and memory processes, along with a complete enumeration of all knowledge dependencies, are not required for many learning tasks. People certainly cannot maintain a complete and consistent knowledge base, neither are they able to perform full dependency-directed backtracking (Stallman & Sussman, 1977) or reason maintenance (Doyle, 1979) for belief revision; rather, they depend on failures of reasoning and memory of past errors to indicate where inconsistencies in their knowledge lie. That is, as knowledge is locally updated, a knowledge base will often become globally inconsistent and partially obsolete. It is at the point in which a system (either human or machine) attempts to reuse obsolete information that inconsistency becomes most apparent and further learning is enabled.[7] People often do know when they err if their conclusions contradict known facts, if plans go wrong, or if they forget (even if they cannot remember the forgotten item). Representations should support such types of self-knowledge, and it is at this level of granularity that an *epistemologically adequate* (McCarthy & Hayes, 1969) content theory of mental representations can be built.

Representing Reasoning Failure to Support Learning

So as to support learning, a theory should have a level of representation that reflects the structure and content of reasoning failures. Cox & Ram (1994) extend the scope of reasoning failure to include the following forms. A failure is defined as an outcome other than what is expected (or a lack of some outcome). If a system incorrectly analyzes some input, or solves some problem incorrectly, so that its expected solution differs from the actual solution given some criteria or feedback, then a failure has occurred. This is the conventional notion of failure and will be termed a *contradiction*. Moreover, if a system expects that it will not be able to compute any answer or the correct answer, but it does nonetheless, then another failure class exists called an *unexpected success*. An *impasse* is defined as either a failure of a process to produce any result or the condition under which no process is available to attempt a result. Alternatively, if a system has no expectation, yet an event occurs which should have been expected, then a *surprise* exists. This paper presents a declarative representation for each of these four classes of reasoning failure.

The basic organization for all of these representations is at the level of a comparison between an expectation and some feedback (either from the environment or additional inference or memory).[8] Oehlmann et al. (in press) stress the importance of metacognitive processing to provide expectations and to monitor comprehension, both in human and

7. See also McRoy (1993) for related discussion of resource constraints on inference, the problems of logical inconsistency and logical omniscience, and the proposed relationship of these problems to the agent's own involvement in introspection.

machine systems. The representations used by any system should support these processes. Although this paper focusses on representation, see Cox & Ram (in press) and Ram & Cox (1994) for their application in an implemented computational system.

Before I examine the representation for reasoning failures, it is worth noting that the basic representation can account for many of the process terms from Figure 1, not just classes of failure. In Figure 4 for example, if the value of the cognize node that produces an expectation, E, is a memory process, the representation can easily capture the distinctions between an incidental reminding, a deliberate recall, and recognition: that is, the three subnodes of "remember" in Figure 1. The structural differences depend on the nodes C and G, and the temporal order of the causal links resulting in nodes E and A (see Table 1). If there is no knowledge goal (Ram, 1991; Ram & Cox, 1994; Ram & Hunter, 1992) to retrieve some memory item, only cues in the environment, and if E is retrieved before A is produced, then the structure is a reminding. On the other hand, if there is a deliberate attempt to a memory item that is later compared to some feedback, A, then the structure represents recall. Finally, if A is presented followed by a memory probe, then the structure represents recognition, whether or not a retrieval goal exists. It is also significant to note that memory "elaboration" of Figure 1can be represented as a feedback loop from E to C such that each new item retrieved adds to the context that enables further memory retrieval.[9]

Table 1: Structural differences between remembering events

Memory Term	Structural Features	Description
Reminding	Has only Cues: E before A	Incidental: No Knowledge Goal
Recall	Cues and Goal: E before A	Deliberate: Has Knowledge Goal
Recognition	May or may not have Goal; A before E	Borderline between 2 above; Has judgement

Contradiction

Figure 5 illustrates the representation for a contradiction failure. Some goal, G, and context or cues, C, enables some cognitive process to produce an expected outcome, E. A subsequent cognitive mechanism produces an actual outcome, A, which when compared to E, fails to meet the expectation. This inequality of actual outcome with expected outcome initiates the knowledge of contradiction. If the cognitive mechanism was some inferential process, then the failure becomes an expectation failure and the node C represents the context, whereas if the process was a memory function, the contradiction is called an incorporation failure and C represents memory cues.[10]

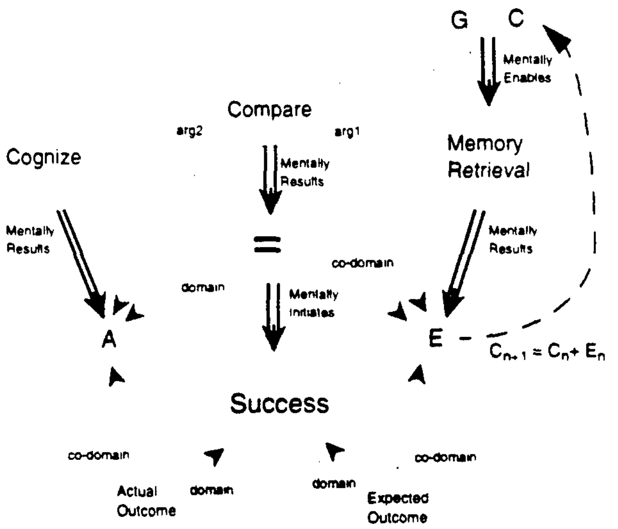


Figure 4. Meta-XP representation of various remembering events
A=actual; E=expected; G=goal; C=context or cues

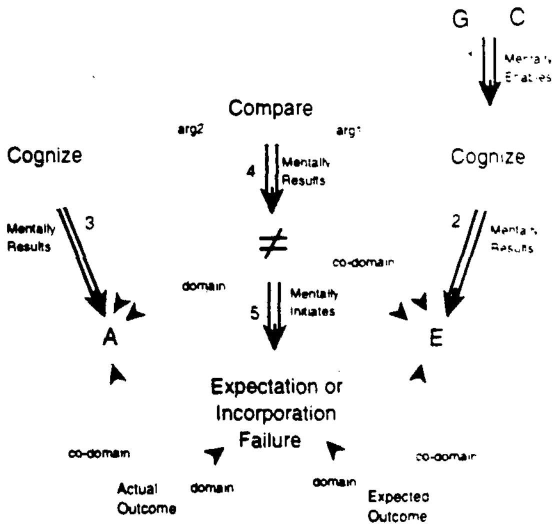


Figure 5. Meta-XP representation of contradiction
A=actual; E=expected; G=goal; C=context or cues

8. See Krulwich (this volume) for another view on basic level categories for mental representations. Instead of grain level, however, his discussion centers on the proper level of abstraction.

9. This characterization is admittedly simplified since cue elaboration incorporates top-down inferential processes as well as bottom-up additions to memory cues.

10. Cox & Ram (1992a) identified four base Meta-XP representations for failure. Two errors of commision are expectation failure and incorporation failure, whereas two errors of omission are retrieval failure and belated prediction. This paper will add the base omission-type representation *construction failure* to this list later in this section.

An incorporation failure occurs when an input concept does not meet a conceptual category. For example, an agent may be told that a deceased individual came back to life (which is false) or a novice student may have a conceptual memory-failure when told that the infinite series $.999\bar{9}$ is equivalent to 1.0 (which is true). These examples contradict the agent's concept of mortal and the naive concept of numbers respectively. Both inferential expectation failure and incorporation failure are errors of commission. Some explicit expectation was violated by later processing or input.

Unexpected success

Figure 6 contains a Meta-XP representation of an unexpected success, a failure similar to contradiction. However, instead of E being violated by A, the expectation is that the violation will occur, yet does not. That is, the agent expects not to be able to perform some computation (e.g., create a solution to a given problem), yet succeeds nonetheless. In such cases the right-most "cognize" term will be some inferential process. If this process is a memory term instead, the failure represents an agent that does not expect to be able to remember some fact on a given memory test, yet at test time or upon further mental elaboration of the cues, the agent remembers it. See for example, the experimental studies of feelings-of-knowing, i.e., judgements of future recognition of an item that was not recalled during some memory test (e.g., Krinsky & Nelson, 1985) and judgements-of-learning, i.e, judgements at rehearsal time as to future memory performance (e.g., Nelson & Dunlosky, 1991). Like the representation of contradiction, the agent expects one outcome (failure), yet another occurs (success) during unexpected successes.

Impasse

Figure 7 represents a class of omission failures that include forgetting as discussed earlier. If the right-most "cognize" term is a memory retrieval process, then the Meta-XP indeed represents forgetting.[11] The impasse is a memory process that fails to retrieve anything. If the node is an inferential process, however, then the impasse failure is equivalent to the failures as recognized by Soar (Newell, 1990) --a blocked attempt to generate the solution to a goal. Thus a *construction failure* is when no plan or solution is constructed by the inference process. In either case, the node E is not in the set of beliefs with respect to the foreground knowledge of the system (i.e., was not brought into or created within working memory).

Surprise

Finally, Figure 8 represents a class of failures rarely treated in any AI system. A surprise occurs when a hindsight process reveals that some expectation was never generated. The explanation is that there was never a goal, G2, to create the expectation, either through remembering or inferring. Some earlier process with goal, G1, failed to generate the subsequent goal. When the node A is generated, however, the system realizes that it is missing. This error, by definition, is a missing expectation discovered after the fact.

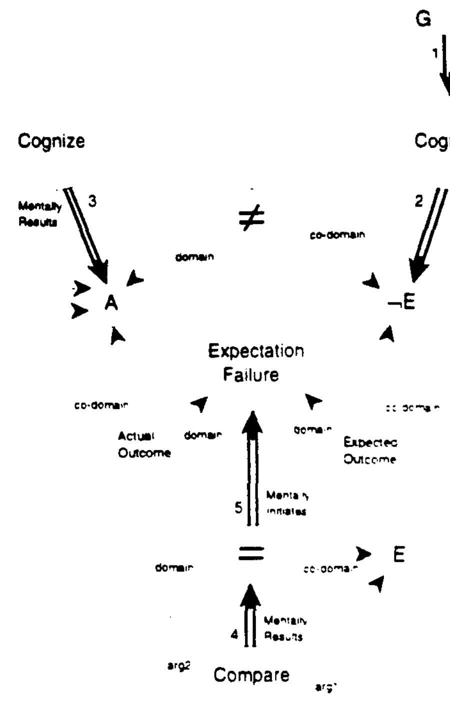


Figure 6. Meta-XP representation of unexpected success
A=actual; E=expected; G=goal; C=context or cues

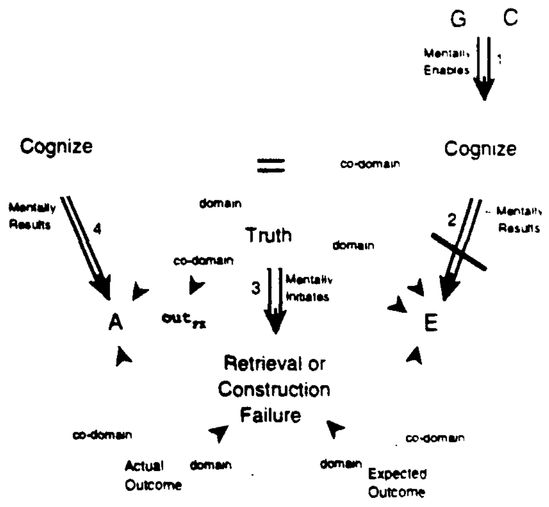


Figure 7. Meta-XP representation of impasse
A=actual; E=expected; G=goal; C=context or cues

11. Compare Fig. 7 with Fig. 4 to see why Forgetting≠ ¬Remember

A Call for Representation

These few examples demonstrate the usefulness of representing mental events and states as well as its complexity. Although difficult to formulate a complete representation of mental events, the effort promises to aid a system when reasoning about itself or other agents, especially when trying to explain why its own or another's reasoning goes astray. Furthermore, even though the CD representation of mental terms leaves much detail unrepresented, the original goal of Schank et al. (1972) to represent the mental domain is a fruitful one. If a representational vocabulary can be fully specified, then these domain independent terms can help many different intelligent systems reason in complex situations where errors occur.

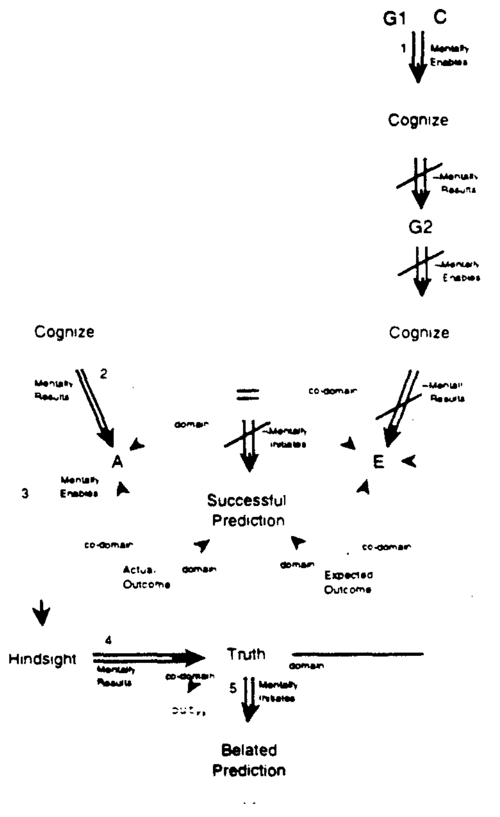


Figure 8. Meta-XP representation of surprise
A=actual: E=expected; G1,G2=goals; C=context or cues

Although many of the details of this preliminary proposal may be overly simplified, it remains an improvement over some of the representational systems proposed in the past such as CD theory. Especially considering the emphasis by Schank et al. on expectation as a driving force in text comprehension and problem solving, the CD representation for the concept of "expectation" is not sufficient to express its central role in cognition. For example, the CD representation for "John expects Bill to become a doctor" (Schank et al., 1972, p. 29) is shown in Figure 9. Very little information is provided in this structure, and few inferences may be obtained from it or learning performed from it.

$$\begin{tikzcd}
\text{Bill} \arrow[d, Leftrightarrow, "f"'] \arrow[r, phantom, "\Longleftrightarrow \text{MLOC (LTM (John))}"] & {} \\
\text{Doctor} &
\end{tikzcd}$$

Figure 9. CD representation of expectation
f=future tense; MLOC=Mental Location; LTM=Long Term Memory

For the above reasons I argue that two primitives are not sufficient to comprise an adequate representational system that can express states and mechanisms of the mental world. Rather, a comprehensive representation needs to be delivered that can be shared between existing formalisms, and where the reuse of representations facilitate both uniformity and transfer across domains in order to support intelligent reasoning, understanding and learning. In support of these goals, I list a number of concepts, dimensions and remaining issues that must be considered in their pursuit:

1. **introspect**, **retrospect**, **inspect**, **reflect**, **suspect**, **expect**.
2. **expect** versus **hope** versus **wish** - depends on knowledge or certainty, that is, one may expect something when confident, but hope for an outcome when doubting, and finally wish for something that is least likely to occur. I suppose we **pray** for the impossible; which leads this list to item 3.
3. **wishful thinking** (in the face of conflicting evidence along with rigid beliefs).
4. **suspend thinking**, **resume thinking** (both opportunistic), **reconsider** (in the light of hindsight, rather than after an interruption usually).
5. **suspend belief**, **(day)dream**, **imagine**, **wonder**. See Schank et al. (1972) p. 18, for a crude representation of wonder; p. 30 for imagine).
6. **apprehension**, **perception**, **precept**, **percept**.
7. **foresee** (foresight), **review** (hindsight), **intuit** (insight).

   Under foresight: **foretell**, **apprehension**, **wish**, **hope, and expect**.

   Under hindsight: **review**, **retrospect**, **remember**, **reconsider**, **reflect**.

   Under insight: **premonition**, **presentiment**, **appre-**

hension, **hunch**, **discovery**, **prescience**.

8. **algorithmic** versus **heuristic** processing (e.g., projective reasoning via possible worlds given different assumptions (logic) versus given different interpretations (analogy or metaphor).
9. **group** (categorize), **compare**, **contrast**, **analogize**.
10. **want**, **need**, **desire**, **goal possession** (all items equivalent).
11. **rehearse**, **elaborate**, **search**; **baffled**, **stumped**, **perplexed**.
12. **notice**, **observe**, **surveil** (search), **discover**, **await** (depends on target or expectation; another scale by deliberation and knowledge explicitness).
13. **intend** (to do an *act*) versus **goal** (to achieve a *state*).
14. **intend**, **attend**, **pretend**, **suspend**, **portend**, **comprehend** (understand). See Schank et al. (1972) pp. 70-75, for an early discussion illuminating the often **incomprehensible** mental term of understand ;-)
15. **(self)explain**.

Acknowledgments

The Air Force Office of Scientific Research supported this research under grant number F49620-94-1-0092. I also thank the Georgia Institute of Technology for its support.

References

Cox, M. T. (1994). Machines that forget: Learning from retrieval failure of mis-indexed explanations. In *Proceedings of the Sixteenth Annual Conference of the Cognitive Science Society* (pp. 225-230). Hillsdale, NJ: Lawrence Erlbaum Associates.

Cox, M. T., & Ram, A. (in press). Interacting learning-goals: Treating learning as a planning task. In M. Keane & J.-P. Haton (Eds.), *Topics in case-based reasoning (Lecture notes in artificial intelligence)*. Berlin: Springer-Verlag.

Cox, M. T., & Ram, A. (1992a). Multistrategy Learning with Introspective Meta-Explanations. In D. Sleeman & P. Edwards (Eds.), *Machine Learning: Proceedings of the Ninth International Conference* (pp. 123-128). San Mateo, CA: Morgan Kaufmann.

Cox, M. T., & Ram, A. (1992b). An explicit representation of forgetting. In J. W. Brahan & G. E. Lasker (Eds.), *Proceedings of the Sixth International Conference on Systems Research, Informatics and Cybernetics: Vol. 2. Advances in Artificial Intelligence - Theory and Application* (pp. 115-120). Windsor, Ontario, Canada: International Institute for Advanced Studies in Systems Research and Cybernetics.

Cox, M. T., & Ram, A. (1994). Failure-driven learning as input bias. In *Proceedings of the Sixteenth Annual Conference of the Cognitive Science Society* (pp. 231-236). Hillsdale, NJ: Lawrence Erlbaum Associates.

Doyle, J. (1979). A Truth Maintenance System, *Artificial Intelligence*, Vol. 12, pp. 231-272.

Fox, S., & Leake, D. (1995). Modeling case-based planning for repairing reasoning failures. In M. T. Cox & M. Freed (Eds.), *Proceedings of the 1995 AAAI Spring Symposium on Representing Mental States and Mechanisms*. Menlo Park, CA: AAAI Press.

Kennedy, A. C. (1995). Using a domain-independent introspection mechanism to improve memory search. In M. T. Cox & M. Freed (Eds.), *Proceedings of the 1995 AAAI Spring Symposium on Representing Mental States and Mechanisms*. Menlo Park, CA: AAAI Press.

Krinsky, R., & Nelson, T. O. (1985). The feeling of knowing for different types of retrieval failure. *Acta Psychologica*, *58*, 141-158.

Krulwich, B. (1995). Cognitive behavior, basic levels, and intelligent agents. In M. T. Cox & M. Freed (Eds.), *Proceedings of the 1995 AAAI Spring Symposium on Representing Mental States and Mechanisms*. Menlo Park, CA: AAAI Press.

Leake, D. (1995). Representing self-knowledge for introspection about memory search. In M. T. Cox & M. Freed (Eds.), *Proceedings of the 1995 AAAI Spring Symposium on Representing Mental States and Mechanisms*. Menlo Park, CA: AAAI Press.

Leake, D. (1992). *Evaluating explanations: A content theory*. Hillsdale, NJ: Lawrence Erlbaum Associates.

McCarthy, J., & Hayes, P. (1969). Some philosophical problems from the standpoint of artificial intelligence. *Machine Intelligence*, 4: 463-502.

McRoy, S. W. (1993). Belief as an effect of an act of introspection: Some preliminary remarks. In *Proceedings of the 1993 AAAI Spring Symposium on Reasoning about Mental States: Formal Theories and Applications*. Menlo Park, CA: AAAI Press.

Oehlmann, R., Edwards, P., & Sleeman, D. (1995). Introspection planning: Representing metacognitive experience. In M. T. Cox & M. Freed (Eds.), *Proceedings of the 1995 AAAI Spring Symposium on Representing Mental States and Mechanisms*. Menlo Park, CA: AAAI Press.

Nelson, T. O., & Dunlosky, J. (1991). When people's Judgements of Learning (JOLs) are extremely accurate at predicting subsequent recall: The "Delayed-JOL Effect." *Psychological Science*, 2(4), 267-270.

Newell, A. (1990) *Unified theories of cognition*. Cambridge, MA: Harvard University Press.

Ram, A. (1991). A theory of questions and question asking. *Journal of the Learning Sciences*, *1*(3&4), 273-318.

Ram, A., & Cox, M. T. (1994). Introspective reasoning using meta-explanations for multistrategy learning. In R. Michalski & G. Tecuci (Eds.), *Machine learning: A multistrategy approach IV* (pp. 349-377). San Mateo, CA: M. Kaufmann.

Ram, A., & Hunter, L. (1992). The use of explicit goals for knowledge to guide inference and learning. *Applied Intelligence*, *2*(1), 47-73.

Schank, R. (1975). *Conceptual information processing*. Amsterdam: North-Holland Publishing.

Schank, R. C. (1986). *Explanation patterns: Understanding mechanically and creatively*. Hillsdale, NJ: LEA.

Schank, R. C., Kass, A., & Riesbeck, C. K. (1994). *Inside case-based explanation*. Hillsdale, NJ: LEA.

Schank, R., Goldman, N., Rieger, C., & Riesbeck, C. (1972). *Primitive concepts underlying verbs of thought*. Stanford Artificial Intelligence Project Memo No. 162, Computer Science Department, Stanford University (NTIS #AD744634)

Stallman, R. M., & Sussman, G. J. (1977). Forward reasoning and dependency-directed backtracking in a system for computer-aided circuit analysis. *Artificial Intelligence*, 9: 135-196.

Stein, G. & Barnden, J. A. (1995). Towards more flexible and common-sensical reasoning about beliefs. In M. T. Cox & M. Freed (Eds.), *Proceedings of the 1995 AAAI Spring Symposium on Representing Mental States and Mechanisms*. Menlo Park, CA: AAAI Press.

Winograd, T. (1985). Frame representations and the declarative/procedural controversy. In R. J. Brachman & H. J. Levesque (Eds.), *Readings in Knowledge Representation* (pp. 358-370). San Mateo, CA: Morgan Kaufmann (originally published in 1975).

To appear in Leake, D.B., editor (in press), *Case-Based Reasoning: Experiences, Lessons, and Future Directions*. AAAI Press

MULTI-PLAN RETRIEVAL AND ADAPTATION IN AN EXPERIENCE-BASED AGENT

Ashwin Ram and Anthony G. Francis, Jr.
College of Computing, Georgia Tech

I. Introduction

The real world has many properties that present challenges for the design of intelligent agents: it is dynamic, unpredictable, and independent, poses poorly structured problems, and places bounds on the resources available to agents. Agents that operate in real worlds need a wide range of capabilities to deal with them: memory, situation analysis, situativity, resource-bounded cognition, and opportunism. In particular, agents need the ability to dynamically combine past experiences to cope with new situations, selecting and merging the relevant parts of remindings to take maximum advantage of their past experience.

To address these issues, we propose a theory of *experience-based agency* which specifies how an agent with the ability to richly represent and store its experiences could remember those experiences with a context-sensitive, asynchronous memory, incorporate the relevant portions of those experiences into its reasoning on demand with integration mechanisms, and direct memory and reasoning through the use of a utility-based control mechanism. We have implemented this theory in the NICOLE multistrategy reasoning system and are currently using it to explore the problem of merging multiple past planning experiences. NICOLE'S control system allows memory and reasoning to proceed in parallel; the asynchronous and context-sensitive nature of that memory system allows NICOLE to return a "best guess" retrieval immediately and then to update that retrieval whenever new cues become available.

But solving the problem of merging planning experiences requires more than just memory and control. We need mechanisms to integrate new retrieved plans into the planner's current reasoning context whenever they are found; to ensure that those new retrieved plans are useful, we need ways to use the planner's current context to generate new cues that can help guide the memory system's search. To solve these subproblems, we have developed the Multi-Plan Adaptor (MPA) algorithm, a novel method for merging partial-order plans in the context of case-based least-commitment planning. MPA allows the merging of arbitrary numbers of plans at any point during the adaptation process; it achieves this by dynamically extracting relevant case subparts and splicing those subparts into partially completed plans. MPA can also help guide retrieval by extracting intermediate goal statements from partial plans.

In this chapter, we briefly review the properties of the real world that present challenges for the design of intelligent agents, examining in particular the need to combine past planning experiences. We then review our theory of experience-based agency and its implementation in the NICOLE system. We present the MPA algorithm, illustrate how it supports the merging of plans at any point during the adaptation process, and describe its foundations in least-commitment case-based planning. We then discuss how MPA can be integrated into various control regimes, including systematic, pure case-based, and interleaved regimes, and describe our implementation of interleaved MPA in NICOLE. We conclude the paper by reviewing other case-based planning work and then outlining our contributions.

2. Exploiting the Past

2.1. The Challenges of the Real World

The real world presents many challenges to an agent, challenges which arise in both artificial domains and in the real-world problems humans face (Bratman, 1987; Hammond, 1989; Orasanu & Connolly, 1993; Pollock, 1995; Sternberg, 1985, 1986; Wooldridge & Jennings, 1995). The real world is *dynamic*, changing independently from the actions of an agent, and it is *unpredictable*, changing in a way too complex for an agent to completely predict. To make things worse, the world and its changes are *relevant* to the agent's goals (so that they cannot be ignored), *sensitive* to the agent's actions (so that the agent cannot act with impunity) and place *resource bounds* on the agent's activity (so that the agent cannot just try everything until it works).

But the real world is also *regular*: patterns exist, encapsulating classes of objects and events and relationships of cause and effect. An agent's ability to effectively use its past experience with these patterns can be key to its successful performance and survival in real-world domains. In this chapter, we take as a given the traditional trappings of a sophisticated intelligent agent, which include *sensors* and *effectors* (Russel & Norvig 1995), *desires* (or values) and *goals* (Bratman, 1987; Pollock, 1995; Schank and Abelson, 1977), *situation analysis* (Kolodner, 1993; Pollock, 1995), *context sensitivity* or *situativity* (Maes, 1990), *efficient resource-bounded cognition* (Kolodner, 1983; Pollock, 1995), and *opportunism* (Hammond 1989; Hayes-Roth & Hayes-Roth, 1979; Simina & Kolodner, 1995). In this context, we will focus on an agent's memory for past experiences and how relevant pieces of multiple past experiences can be effectively integrated during problem solving in order to synthesize solutions for new problems.

2.2. The Need to Combine Experiences

Taking advantage of past experiences is the foundation of case-based reasoning. When confronted with a problem, a case-based reasoner recalls a past experience and adapts it to provide the solution to the new problem. Unfortunately, in many real-world domains we cannot count on a single past experience to provide the outline of a solution to our problems. For example:

- A graduate student asked to present his first paper at an overseas conference must draw on separate past experiences in preparing talks for conferences within his country and preparing his passport and flight arrangements for vacations outside of his country.
- A host planning his first large dinner party must recall both the outline of a menu as served at family gatherings and his separate experiences at preparing individual dishes for himself.
- A home hobbyist attempting his first large piece of furniture must recall both past examples of that type of furniture to provide a design and

experiences with acquiring, assembling and finishing individual components.'

All of these problems have something in common: every *piece* of the solution can be constructed entirely out of the agent's past experience (with suitable adaptation), but no *single* past experience suffices to solve the entire problem. For these types of problems, unless a case-based reasoning system has the ability to combine several past experiences, it will have to resort to expensive from-scratch reasoning in order to solve the problem.

Some CBR planning systems combine multiple cases during reasoning. However, they either gather all partial plans at retrieval prior to adaptation (e.g., PRODIGY/ANALOGY, Veloso 1994; Chapter 8, this volume), break plans into *snippets* at storage time so they can be retrieved individually (e.g., CELIA, Redmond 1990, 1992), or combine cases recursively, applying complete past cases to sub-problems within a larger problem (e.g., ROUTER, Goel et al., 1994; SBR, Turner, 1989; PRODIGY/ANALOGY, Veloso, 1994; Chapter 8, this volume). None of these approaches is entirely satisfactory, for various reasons.

It is not entirely clear that all of the knowledge needed to solve a problem can be assembled at the beginning of problem solving. For example, in the furniture example, it is not entirely clear whether or not the agent needs to buy new sandpaper, and hence unclear whether the agent should recall past experiences of buying sandpaper at a hardware store. This uncertainty arises out of several concerns: the uncertainty of the world state (how much sandpaper does the agent have?), uncertainty in the effectiveness of agent actions (how much wood will a piece of sandpaper sand?) and the potential of exogenous events that can invalidate parts of the plan (if a friend drops and scars a piece of the furniture, will the agent have enough sandpaper to remove the scar, or will he need to buy more?). But it can also arise out of *the plan itself:* until the agent has decided on a design for the piece and how much wood will be involved, it is unclear precisely how much sandpaper is needed, and hence unclear whether or not a plan should be retrieved. If some amount of sandpaper is on hand, the goal of acquiring sandpaper may not even arise until late in the planning process, when it has become clear that the amount on hand is insufficient.

Precomputing case snippets also has drawbacks. While this allows us to extract subparts of a case to meet the needs of an open goal in a plan, these subparts need to be computed at storage time. Unfortunately, it is not clear that every useful breakdown of a case can be computed in advance. For example, in the foreign conference example, the two past experiences must be closely interleaved to produce a new plan. If the agent has not stored the passport experience as a separate snippet, the agent may not be able to extract that particular piece of a case if it is retrieved — if the agent was able to retrieve it at all.

Recursive case-based reasoning — using case-based reasoning to satisfy subgoals in a partial plan — is an effective way to combine multiple cases because information about the partially constructed solution can be used to help select additional cases to complete the solution. However, existing systems that use recursive case-based reasoning, such as ROUTER and PRODIGY/ANALOGY, can only retrieve whole cases and attempt to apply them to the subproblems at hand; these systems tend to fill in one 'gap' at a time and neither search for nor attempt to use cases that might fill several gaps in the plan at once, even if such cases exist.

We advocate a more flexible approach: using more complete information about the *current* state of a partially adapted plan to guide the retrieval of cases which address as many deficiencies in the plan as possible, and dynamically selecting the relevant portions of those cases to integrate with

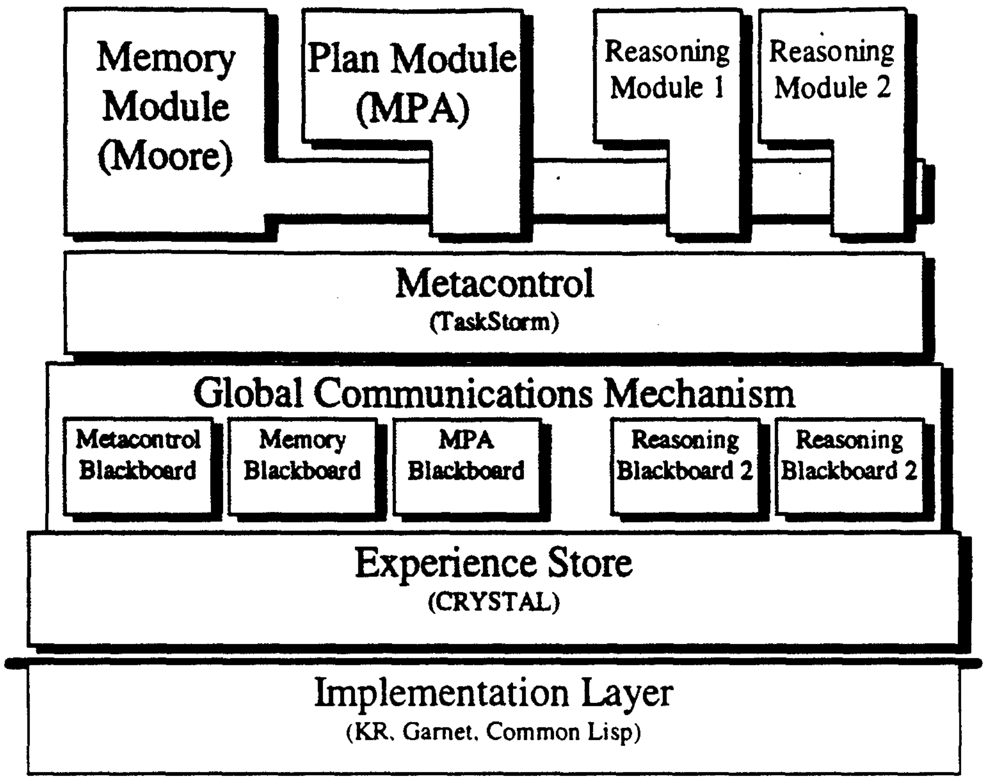


Figure 1. The Architecture of NICOLE.

our current reasoning process. We believe that the key to achieving this is not to attempt to solve this problem in isolation, but instead to look at how the design of a complete agent could provide us with the tools with which to solve the problem.

3. An Architecture for Real-World Domains

3.1. Experience-Based Agency

A growing community of AI researchers has come to believe that successfully meeting the challenges posed by the real world will require building comprehensive agent architectures, rather than by tackling individual problems separately and trying to combine the solutions after the fact (e.g., Anderson, 1983; Hayes-Roth, 1995; Newell, 1990; Nilsson, 1995; Pollock, 1995). We subscribe to this view; in particular, we believe that the problem of deciding when and what cases to retrieve and how to integrate those cases into the system's current plans can only be solved in the context of the retrieval and reasoning needs of an entire agent functioning in a real-world domain.

To explore this problem, we have developed a theory of *experience-based agency*, which specifies a class of agent capable of not only integrating experiences on demand but also retrieving those experiences in a context-sensitive and asynchronous fashion. The theory has five primary components. The core components are a richly represented *experience store* and a *global communications mechanism*, which together lay the foundation for a *context-sensitive asychronous memory system*. To cope with the potential retrieval of new information at any time, reasoning mechanisms are equipped with *integration mechanisms*; to coordinate reasoning and memory, the theory proposes a central *metacontroller*. Figure 1 illustrates our current implementation of these components in the NICOLE[1] multistrategy reasoning system.

3.2. Memory and Control in an Experience-Based Agent

An experience-based agent is designed to operate in a dynamic, unpredictable world with limited information, and as such naturally requires the ability to combine multiple experiences on demand, clip out their irrelevant subparts, and splice them together into a complete solution for the problem at hand. Driving this integration of experience is the asynchronous retrieval of relevant experiences by the independent memory system. In a dynamic world, we cannot guarantee that the cues and specifications that reasoning provides to memory will be sufficient to allow retrieval of the best experiences quickly enough to allow retrieval to continue uninterrupted; an experience-based agent avoids this problem by allowing memory to return a "best guess" initially while continuing to search memory in parallel with any reasoning, acting, or sensing operations being performed by the agent.[2] Figure 2 illustrates the "life history" of a typical parallel memory search.

The "rich" knowledge representation used in an experience-based agency system's experience store (implemented in NICOLE as a highly connected semantic network[3]) allows the memory to make connections between reasoning operations and past cases; the global communications mechanism (implemented in NICOLE as a set of task-specific blackboards) ensures that the content of reasoning operations is visible to the memory to make it context-sensitive. In contrast, where knowledge representation and the global communications mechanism aim to increase memory's ability to retrieve, the metacontroller limits it: it provides utility metrics which limit when memory (or other reasoning modules[4]) can execute. If no sufficiently suitable retrieval can be found, metacontrol ensures that the agent spends its time processing the information it already has, rather than allow the memory to return every partial match, no matter how slight.

3.3. Reasoning in an Experience-Based Agent

However, while these components are necessary to achieve the desired behavior, they are not sufficient; modification must be made *within* reasoning and memory as well. Traditional reasoning algorithms have well-defined inputs and outputs, and it simply does not make any sense to say that "new information can be retrieved at any time" without some *integration mechanism* that can incorporate that information into reasoning. Similarly, traditional retrieval algorithms operate at

[1] NICOLE is named after a sentient computer in a science fiction short story by one of the authors (Francis 1995).

[2] Because of its ability to return a best guess, an experience-based agent has many similarities to the interactive "shoot-first" case-based reasoning systems advocated by Riesbeck (1993). Given a problem input by the user, a shoot-first system attempts to quickly retrieve an approximate set of cases and/or propose a sketchy solution, and then uses feedback from the user to redefine the problem, refine the search, or adapt the solution. Both experience-based agents and shoot-first systems require similar memory capabilities; however, the primary "user" of an experience-based agent's quick retrievals is the agent itself.

[3] This semantic network is "highly connected" in two senses: both in the network structure (as in Kilmesch, 1994) and in the semantics of the network, in which all nodes and links are reified concepts (as in Brachman, 1985; Wilensky, 1986).

[4] Reasoning modules are implemented in NICOLE as supertasks (Moorman & Ram, 1994) which combine characteristics of task-specific blackboard panes (Hayes-Roth & Hayes-Roth, 1979) and extended Reactive Action Packages (RAPs; Firby, 1989).

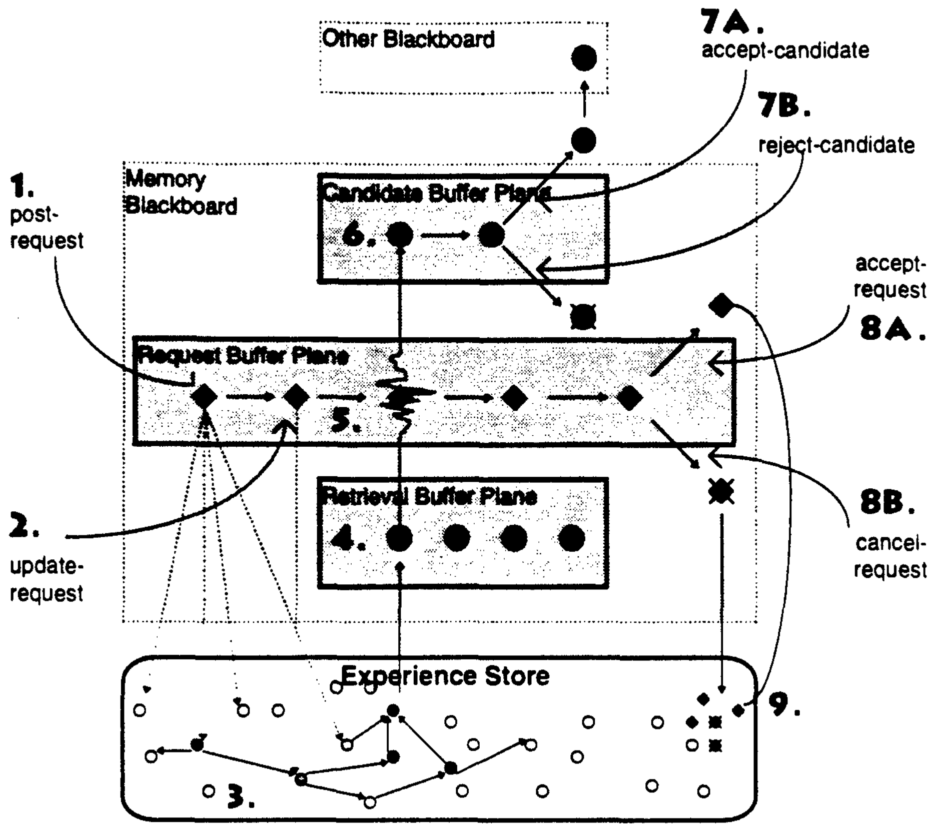


Figure 2. The Life History of a Retrieval in NICOLE.

1. A retrieval begins when a reasoning module calls **post-request**, which adds a new **request-node** (symbolized by a diamond) to the **request buffer** of the memory blackboard. **2.** A request-node may be updated with a **update-request** call at any time. **3.** Each time a request is posted or updated (or when activity occurs in other system blackboards) **activation** spreads to nodes in long term memory (the **experience store**). **4.** On every retrieval cycle, a limited number of **active nodes** (symbolized by dark circles) are retrieved to the **retrieval buffer** for consideration. **5.** Each pending request in the request buffer is **matched** against the candidates in the retrieval buffer. **6.** Successful matches are posted to the **candidate buffer** to be copied to the requesting blackboard or process. **7.** A retrieval candidate can be accepted or rejected by the **accept-candidate** and **reject-candidate** calls, which update the state of the request to allow it to more accurately select future matches. **8.** The reasoning module may at any time decide to terminate processing of a request by accepting it through an **accept-request** call or rejecting it with a **cancel-request** call. **9.** Terminated requests, both successful and unsuccessful, are stored in long term memory and used by the **storage module** (not shown) to adjust associative links in long term memory and retrieval parameters in the matching and candidate retrieval systems.

the command of the reasoning system, performing each search of memory separately from every other and basing each search only on the cues and specifications provided at the particular moment the search was initiated. In order to take maximum advantage of an independent memory we must allow it to establish *knowledge goals* which are based on the needs of reasoning but which can be independently pursued, updated, and (eventually) satisfied.

3.4. Exploiting Past Plans in an Experience-Based Agent

In addition to the general need to combine past experiences into a coherent solution to its current problems, if an experience-based agent is called upon to perform planning tasks, it needs the specific capability to combine multiple *plans* on demand, clip out their irrelevant subparts, and splice them together into a complete plan. To solve this problem, we have configured NICOLE to implement an interactive multi-plan adaptation system (called NICOLE-MPA).

NICOLE-MPA represents an advance over traditional case-based reasoning systems in two ways. First, NICOLE-MPA uses a novel algorithm called the Multi-Plan Adaptor (MPA) to extend the concept of multi-plan adaptation to the least-commitment case-based planning framework. Within the context of the NICOLE system, the MPA algorithm also provides both an integration mechanism and a knowledge goal generation mechanism for least-commitment planners using partially ordered plans. Second, NICOLE-MPA provides a framework for the study of various heuristics for multi-plan adaptation.

To set the stage for NICOLE-MPA, we will discuss the least-commitment planning framework and its case-based implementation in systems like SPA, then detail the MPA algorithm itself and how it extends the traditional systems upon which it is founded. Then, we will discuss how NICOLE is configured to implement interactive multi-plan adaptation, and conclude by discussing the potential efficiency gains and hazards of NICOLE-MPA.

4. Interactive Multi-Plan Adaptation

4.1. An Overview of Case-Based Least Commitment Planning

Least-commitment planning departs from traditional planning systems by delaying decisions about step orderings and bindings as much as possible to prevent backtracking (Weld, 1994). Least-commitment planners solve problems by successive *refinement* of a partial plan derived from the initial and goal conditions of the problem. Plans are represented as sets of steps, causal links between steps, variable bindings and ordering constraints. Beginning with a skeletal partial plan based on the initial and goal conditions of the problem, a least-commitment planner attempts to refine the plan by adding steps, links and constraints that eliminate open conditions or resolve threats.

The Systematic Plan Adaptor algorithm (Hanks & Weld, 1995) is a least-commitment algorithm for case-based planning. SPA is based on four key ideas: treat adaptation as a *refinement* process (based on the generative least-commitment planner SNLP; McAllester & Rosenblitt, 1991), annotate partial plans with *reasons* for decisions, add a *retraction mechanism* to remove decisions, and add a *fitting mechanism* to fit previous plans to current situations. Reasons support the fitting and retraction mechanisms by allowing SPA to determine the dependencies between steps. Once a plan has been retrieved, it may contain steps and constraints that are extraneous or inconsistent with the new situations; during fitting, reasons allow SPA to identify extraneous steps and remove them; during adaptation, reasons allow SPA to isolate steps which are candidates for retraction.

One limitation of SPA is that it is a single-plan adaptor; even if a new problem could be solved by merging several plans, SPA must choose only one and adapt it to fit. This limitation arises from SPA's attempt to maintain *systematicity* and *completeness*. A systematic planner never repeats its work by considering a partial plan more than once; a complete planner always finds a solution if it exists. SPA ensures these properties (in part) by choosing only one refinement at a time, by never retracting any refinement made during adaptation (to avoid reconsidering plans), and by adding all possible versions of any refinement it chooses (to avoid missing a solution). A full discussion of completeness and systematicity in SPA is beyond the scope of this paper (but see Francis & Ram, 1995; Hanks & Weld, 1995).

4.2. Multi-Plan Adaptation

Because it adapts only one plan, SPA can resort to significant amounts of from-scratch planning even when the knowledge needed to complete the plan is present in the case library. To make the most effective use of the planner's past experience, we need the ability to recognize when a partial plan needs to be extended, select plans to that address the deficiency, and then extract and merge the relevant parts of the retrieved plan into the original plan.

MPA resolves this problem in SPA by allowing the retrieval and merging of arbitrary numbers of cases at any point during the adaptation process. MPA also allows the dynamic extraction of relevant parts of past cases. To achieve this, we extended the SPA framework by adding three key mechanisms:

- a goal deriver, which extracts *intermediate goal statements* from partial plans
- a *plan clipper*, which prepares plans for merging using a modified plan fitting mechanism
- a *plan splicer*, which merges two plans together based on their causal structure

Intermediate goal statements are MPA's knowledge goals; they provide MPA with the ability to merge partial plans at any point of the adaptation process and contribute to its ability to dynamically extract the relevant subparts of retrieved cases. Intermediate goal statements are extracted by the inverse of a representational "trick" that SPA uses to construct a skeletal plan if it can't find a relevant case in its library. The trick is simple: build a skeletal plan by adding dummy initial and final steps whose post- and pre-conditions match the initial and goal conditions of the problem. This technique is also used in SPA's generative predecessor SNLP (McAllester & Rosenblitt, 1991) as well as a host of other STRIPS-based planning systems. MPA inverts this trick by extracting new goals from the open conditions of a partial plan. As planning proceeds, open conditions in the goal statement are resolved, but new open conditions are posted as new steps are added. MPA constructs an intermediate goal statement by extracting these new open conditions and using them to form the new goal state, and by extracting the initial conditions of the partial plan can be extracted and using them to form the new initial conditions.[5]

[5] Unfortunately, since ordering constraints and binding constraints may be posted to the plan at any time, only the initial conditions can be guaranteed to be valid conditions for the intermediate goal statement. Conditions established by other steps of the plan may be clobbered by the addition of new steps and new ordering constraints. However, it might be possible to develop heuristics that select additional initial conditions that are likely to hold, perhaps in conjunction with more complex retrieval, fitting and splicing algorithms. In general, deciding which parts of a plan can be extracted to form a sensible and effective intermediate goal statement is a difficult and unsolved problem.

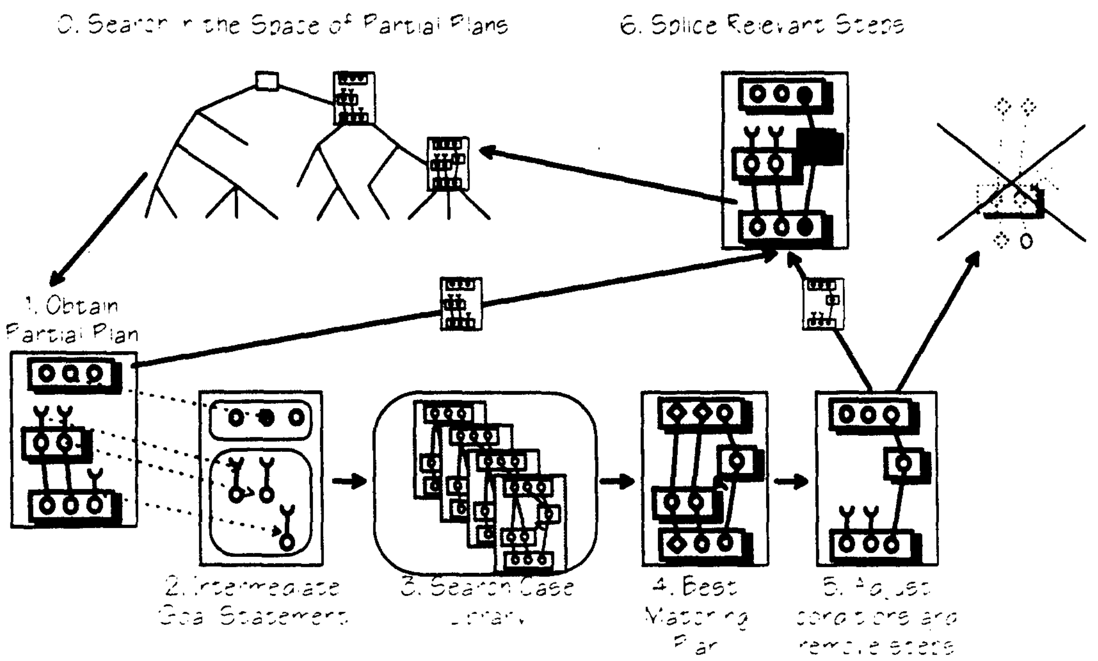

0. Search in the Space of Partial Plans
6. Splice Relevant Steps
1. Obtain Partial Plan
2. Intermediate Goal Statement
3. Search Case Library
4. Best Matching Plan
5. Adjust conditions and remove steps

Figure 3. Overview of Multi-Plan Adaptation

Multi-plan adaptation begins when **1**. a partial plan is obtained, either directly from a goal statement, from an initial case fitting, or from ongoing adaptation processes. **2**. An intermediate goal statement is extracted from the plan, consisting of the initial conditions known to be true in the world and the set of preconditions not yet satisfied in the plan. **3**. The case library is searched for a matching plan in exactly the same way that it is searched for initial case fittings. **4**. The best matching plan is retrieved and **5**. is adjusted to have the right set of initial and goal conditions and to remove extraneous plan steps. **6**. The steps are recursively spliced into the plan, beginning with the links that match to the intermediate goal statement and then moving backwards through the plan along the paths of the causal links. **0**. Finally, the successfully spliced plans are returned for further adaptation or splicing.

Just like the original goal statement, the intermediate goal statement can be used to retrieve and fit a partial plan. However, the result of this process is not a complete fitted plan suitable for adaptation; it is a *plan clipping* that satisfies some or all of the open conditions of the partial plan from which the intermediate goal statement was derived. To take advantage of the plan clipping for adaptation, it must be *spliced* into the original partial plan. Together, plan clipping and splicing form MPA's integration mechanism for incorporating new plans into the current reasoning context (Figure 3).

Our splicing mechanism uses the intermediate goal statement to produce a mapping between the partial plan and the plan clipping, pairing open conditions from the partial plan with satisfied goal conditions from the plan clipping. The plan splicer uses this mapping to perform a guided refinement of the original partial plan, selecting goal conditions from the clipping and using the links and steps that satisfied them as templates to instantiate similar steps and links in the original plan. As these steps are added, new mappings are established between open conditions in the new steps and satisfied preconditions in the clipping and are added to the queue of mappings that the splicer is processing. Hence, the plan splicer performs a backwards breadth-first search through the causal structure of the plan clipping, using links and steps in the clipping to guide the instantiation of links and steps in the original plan. Figure 4 briefly outlines the MPA algorithm.

```
Input: A partial plan P, and a case library C.
Output: A new partial plan P'.

procedure MPA (P, C):
1 P' ← Copy-Plan(P)
2. igs ← GetIntermediateGoalStatement(P')
3. plan ← RetrieveBestPlan (C, igs)
4. {clipping, mapping} ← FitPlan (plan, igs)
5. for cgp in mapping do
6.    if Producer-Exists (oc-gl-pair, P')
7.          then  Splice-Link (oc-gl-pair, P', clipping)
8.          else  Splice-Step (oc-gl-pair, P', clipping)
9.    AddNewOpenCond-GoalPairs(mapping, P')
10. return P'
```

Figure 4. The MPA Algorithm

4.3. Controlling Multi-Plan Adaptation

Merely having the ability to splice plans together does not allow us to take advantage of past experience. We need to decide what experiences to combine and when to combine them. Because the MPA algorithm can potentially be performed at any point during the adaptation process — using an initial skeletal plan derived from the initial and goal statement, using a fitted plan derived from retrieval, or using an adapted plan after some arbitrary amount of retraction and refinement — we have considerable flexibility in deciding what to retrieve, when to retrieve it and when to merge it.

We have considered three alternative control regimes, each of which makes different commitments about when to retrieve and when to adapt. On one end of the spectrum, *Systematic MPA* preserves SPA's property of systematicity by splicing all retrieved cases before (generative) adaptation begins. On the other, *Extreme MPA* never performs generative adaptation and instead uses a set of *pivotal cases* (Smyth & Keane, 1995) to guarantee completeness.

Both Systematic and Extreme MPA make extreme commitments: either integrate all knowledge before generative adaptation begins, or never generatively adapt and rely solely on past experience. An alternative approach is to allow plan splicing at any point during adaptation. In the middle stands *Interactive MPA*, a control regime that can potentially attempt a retrieval at any time, either with the initial skeletal plan or with partial plans produced as a result of adaptation. Since the results of splicing cause large jumps in the search space, this regime deliberately departs from the systematicity of SNLP and SPA in an attempt to solve the problem with less search.

However, allowing arbitrary plan retrieval and plan splicing is not without cost. Performing a full search of the system's case library at every step of the problem space could be computationally prohibitive. The costs of searching the case library at every step of the problem space could outweigh the benefits of reduced search, especially if the system enters a "slump" — an adaptation episode which begins and ends with the application of relevant clippings, but which goes through a series of intermediate plans for which the system cannot match any existing plans in its case library. Clearly, it is worthwhile to retrieve and apply clippings at the beginning and end of a slump, but a full search of the case library at each intermediate step could cost more than the benefits that the initial and final retrievals provide. This is the *swamping utility problem* — the benefits of case retrieval can be outweighed by the costs of that retrieval, leading to an overall degradation in performance as a result of case learning (Francis & Ram, 1995).

4.4. Interactive Multi-Plan Adaptation

Developing heuristics for deciding when and when not to retrieve is a challenging open problem. The experience-based agency theory suggests that plan adaptation should be driven by retrieval; to test this theory, we have implemented an Interactive MPA system within a specially configured version of NICOLE, called NICOLE-MPA.

NICOLE-MPA is an instantiation of NICOLE in which all of MPA's temporary data structures are implemented using NICOLE blackboards. Case adaptation, intermediate goal statement generation, plan clipping, and plan splicing are all implented as NICOLE task modules, which run in parallel with the memory module and other modules of the NICOLE system. Because of this parallelism, NICOLE-MPA can potentially attempt a retrieval at any time, either with the initial skeletal plan or with partial plans produced as a result of adaptation. To support this retrieval, goals, plans and intermediate goal statements are augmented with *plan tags*, which allow the smooth integration of traditional least-commitment planning structures with the richly interconnected semantic network which makes up NICOLE-MPA'S experience store. The plan tagging mechanism allows NICOLE-MPA to manipulate intermediate goal statements, plans and goals in a completely native fashion while providing the memory system with the cues necessary to continue to refine retrieval.

A typical problem-solving session in NICOLE-MPA begins with the presentation of a problem. From this problem, NICOLE-MPA generates a goal statement and uses the goal statement to post a retrieval request. If some partially matching case can be found, the system returns it as its *current* best guess; however, the retrieval request remains active. As the system adapts the plan (using a specially "wrapped" and modified version of the SPA algorithm incorporated into a NICOLE task module), it generates plan tags for each partial plan it generates, providing additional cues for the memory module to attempt further retrievals. When memory finds a new past case whose degree of match exceeds a certain threshold, NICOLE-MPA's metacontroller allows memory to return it as a new guess. If it is applicable, the metacontroller may schedule the plan splicing module, which will integrate it into the current partial plan.

4.5. The Benefits of Multi-Plan Adaptation

Both adapting a single partial plan and adapting merged partial plans can produce significant benefits over generative problem solving. The cost of generative planning is exponential in the size of the final plan produced, whereas fitting a plan is a linear operation in the size of the plan. Hence, the potential exists for substantial improvement through retrieval and adaptation if an appropriate past plan exists, especially for large plans. In certain domains, SPA has demonstrated significant improvements over generative planning. However, if large gaps exist in the retrieved partial plan, SPA must resort to adaptation, which, like generative problem solving, has an exponential cost in the number of steps that need to be added.

While this amount of adaptation may be a significant improvement over complete from-scratch problem solving, the potential exists to reduce that even further by using MPA to clip and splice more partial plans. Fitting a clipping and splicing a clipping are linear operations in the size of the plan being spliced. Hence, the potential exists for substantial improvement through plan merging if an appropriate past plan exists, especially if the gaps in the existing plan are large. An initial implementation of MPA for a test domain indicated significant speedups (beginning at 30%) over SPA for even the smallest examples (solution size of the final plan = 5 steps).

5. Related Work

There are wide bodies of work on both least-commitment planning and case-based reasoning. The most relevant example of that work to this research is of course SPA, upon which MPA builds. Other similar plan reuse systems include PRIAR (Kambhampati & Hendler 1992) an SPA-like system based on NONLIN, and XII (Golden et al 1994), an SPA-like system that plans with incomplete information. Hanks and Weld (1995) discuss these and other plan reuse systems from the perspective of the SPA framework.

MPA's plan splicing mechanism is in many ways similar to DERSNLP (Ihrig & Kambhampati 1994), a derivational analogy system built on top of SNLP that uses *eager replay* to guide a partial order planner. While DERSNLP's eager replay mechanism is in some ways similar to a limiting case of Systematic MPA in which a single plan is retrieved and spliced into a skeletal plan derived from an initial problem statement, DERSNLP goes beyond SPA's reason mechanism and includes a full derivational trace of problem solving in its cases. While DERSNLP and its extension DERSNLP-EBL focus on when it is profitable to retrieve a partial plan, unlike NICOLE-MPA they do not provide the capability of interrupting adaptation as a result of an asynchronous memory retrieval, nor do they provide the ability to integrate the results of multiple plans.

Combining multiple plans in case-based reasoning is not a new idea. The PRODIGY/ANALOGY system (Veloso 1994; Chapter 8, this volume) can retrieve and merge the results of an arbitrary number of totally ordered plans during the derivational analogy process. However, because PRODIGY/ANALOGY manipulates and stores totally ordered plans, it runs into significant issues on deciding how to interleave steps (Veloso, 1994, p124-127), an issue MPA avoids because of its least-commitment heritage. Furthermore, PRODIGY/ANALOGY deliberately limits its capability to retrieve and combine cases on the fly in an attempt to reduce retrieval costs.

The ROUTER path-planning system (Goel et al., 1994) has the ability to recursively call its case-based methods to fill in gaps in a planned route when no exactly matching case can be found. (PRODIGY/ANALOGY has a similar capability to call itself recursively, although the full implications of this ability have not yet been explored). Like NICOLE-MPA, ROUTER has the ability to combine multiple cases, although only in a synchronous fashion because its memory does not support spontaneous retrieval. However, each of these cases must be complete cases; it does not have the ability to clip out the relevant portion of a case at retrieval time. ROUTER does have the ability to break a case up into subcases at storage time, but the results show that computational costs of this storage computation outweigh the benefits in improved retrievals (Goel et al. 1994, p. 63).

The JULIA system (Hinrichs 1992) also has the ability to combine pieces of several past cases, but this is largely a domain-dependent algorithm for merging declarative structures, rather than a domain independent planning system. The CELIA system (Redmond 1990, 1992) stores cases as separate *snippets*, case subcomponents organized around a single goal or set of conjunctive goals. Snippets provide CELIA with the ability to retrieve and identify relevant subparts of a past case based on the system's current goals. Note that while snippets are superficially similar to plan clippings, plan clippings are constructed dynamically during problem solving, whereas snippets need to be computed and stored in advance.

Clippings are similar to macro operators (Fikes et al. 1972) in that they use past experience to combine several problem solving steps into a single structure that can be applied as a unit, allowing the system to make large jumps in the problem space and avoid unnecessary search. However, macro operators differ from clippings in two important ways. First, macro operators are traditionally precomputed at storage time, whereas clippings are computed dynamically; second, macro operators are fixed sequences of operators, whereas clippings are partially ordered sets of operators that may be resolved in a wide variety of ways in the final plan.

Kambhampati & Chen (1993) built and compared several systems that retrieve partially ordered case-like "macro operators". They demonstrated that least-commitment planners could take greater advantage of past experience than totally-ordered planners because of their ability to efficiently interleave new steps into these "macro-operators" during planning. While this work focuses primarily on interleaving new steps into single past plans, the explanations the authors advance for the efficiency gains they detected could be extended to suggest that least-commitment planners would be superior to totally-ordered planners when interleaving multiple plans. The NICOLE-MPA system should provide us a testbed with which we can empirically evaluate this hypothesis.

6. Conclusion

We have presented the Multi-Plan Adaptor, an algorithm that allows a case-based least-commitment planner to take advantage of the benefits of several past experiences. MPA provides the ability to retrieve and merge dynamically selected case components at any point during the adaptation process by extracting an intermediate goal statements from a partial plan, using the intermediate goal statement to retrieve and clip a past plan to the partial plan, and then splicing the clipping into the original partial plan.

Multi-plan adaptation has the potential for substantial speedup over single-plan adaptation, but in order for those benefits to be realized MPA must be embedded within a control regime that decides when the system attempts a retrieval, when the system merges, and when the system resorts to adaptation. We have used the NICOLE multistrategy reasoning system to implement an interactive control regime in which cases may be retrieved at any point during adaptation. To cope with the potentially swamping cost of retrieval at every adaptation step, NICOLE-MPA employs an asychronous, resource-bounded memory module that retrieves a "best guess" and then continues to monitor the progress of adaptation, returning a new or better retrieval as soon as it is found.

We believe that the ability to combine multiple plans and the ability to perform asynchronous retrievals form integral parts of any complete agent that functions in a complex domain; moreover, actually implementing these capabilities in an efficient and sensible way can only be done by considering the architecture of a complete agent. Combined with a metacontroller sufficient to make them work together, multi-plan adaptation and asynchronous retrieval form the cornerstone of our theory of experience-based agency, a theory of how an agent could take maximum advantage of its past experiences to cope with the problems of the real world.

References

Anderson, John R. (1983). *The Architecture of Cognition.* Cambridge, Massachusetts: Harvard University Press.

Brachman, R. J. (1985). An Overview of the KL-ONE Knowledge Representation System. Cognitive Science, 9, (1985) 171-216

Bratman, M. (1987). *Intentions, Plans and Practical Reason*. Harvard.

Fikes, Hart and Nilsson. (1972). STRIPS. *Artificial Intelligence*, 1972, pp. 189-208.

Firby, J. (1989). Adaptive Execution in Complex Dynamic Worlds Ph.D. Thesis, Yale University Technical Report, YALEU/CSD/RR #672, January 1989.

Francis, A. and Ram, A. (1995). A Comparative Utility Analysis of Case-Based Reasoning and Control-Rule Learning Systems. In *Proceedings, ECML-95*, Heraklion, Crete, 1995.

Francis, A.G. (1995). "Sibling Rivalry." *The Leading Edge: Magazine of Science Fiction and Fantasy*, 30, February 1995, pp. 79-102.

Goel, A.K., Ali, K.S., Donnellan, M.W., Garza, A.G., and Callantine, T.J. (1994). Multistrategy Adaptive Path Planning. *IEEE Expert (9) 6*, December 1994, pp. 57-65.

Golden, K., Etzioni, O., & Weld, D. (1994). Omnipotence Without Omniscience: Sensor Management in Planning. In *Proceedings of AAAI-94*. AAAI Press/MIT Press.

Hammond, K. (1989). Opportunistic Memory. In *Proceedings of the 1989 Meeting of the International Joint Committee on Artificial Intelligence*.

Hanks, S., & Weld, D. (1995). A Domain-Independent Algorithm for Plan Adaptation. *Journal of Artificial Intelligence Research 2*, pp. 319-360.

Hayes-Roth, B. (1995). Agents on stage: Advancing the State of the Art of AI. *Knowledge Systems Laboratory Report No. KSL 95-50*. May 1995, Knowledge Systems Laboratory, Stanford University.

Hayes-Roth, B., & Hayes-Roth, F. (1979). A cognitive model of planning. *Cognitive Science (2)*, pp. 275-310.

Hinrichs, T.R. (1992). *Problem Solving in Open Worlds: A Case Study in Design*. Lawrence Erlbaum.

Ihrig, L. & Kambhampati, S. (1994). Derivation Replay for Partial-Order Planning. In *Proceedings of AAAI-94*. AAAI Press/MIT Press.

Kambhampati, S. & Chen, J. (1993). Relative Utility of EBG based Plan Reuse in Partial Ordering vs. Total Ordering Planning. In *Proceedings of AAAI-93*. AAAI Press/MIT Press.

Kambhampati, S. & Hendler, J. (1992). A Validation Structure Based Theory of Plan Modification and Reuse. *Artificial Intelligence, 55*, 193-258.

Klimesch, W. J. (1994). *The structure of long-term memory: A connectivity model of semantic processing*. LEA.

Kolodner, J.L. (1993). *Case-based Reasoning*. Morgan Kaufmann.

Maes, P. (1990). Situated agents can have goals. In. P. Maes, Ed., *Designing Autonomous Agents: Theory and Practice from Biology and Engineering and Back*. MIT.

McAllester, D., & Rosenblitt, D. (1991). Systematic Nonlinear Planning. In *Proceedings of the Ninth National Conference on Artificial Intelligence*, pp. 634-639. AAAI Press/MIT Press.

Moorman, K. & Ram, A. (1994). Integrating Creativity and Reading: A Functional Approach. In *Proceedings of the Sixteenth Annual Conference of the Cognitive Science Society*, Atlanta, GA, August 1994.

Newell, A. (1990). *Unified theories of cognition*. Harvard.

Nilsson, N. (1995). Eye on the prize. *AI Magazine (16) 2*, Summer 1995, pp.9-17.

Orasanu, J. & Connolly, T. (1993). The Reinvention of Decision Making. In Klein, G. A., Orasanu, J., Calderwood, R., and Zsambok, C.E., eds., *Decision Making in Action: Models and Methods*. Ablex.

Pollock, J.L. (1995). *Cognitive Carpentry: A blueprint for how to build a person*. MIT Press.

Redmond, M. (1990). Distributed Cases for Case-Based Reasoning: Facilitating Use of Multiple Cases. In *Proceedings of AAAI-90*. AAAI Press/MIT Press.

Redmond, M. (1992). Learning by Observing and Understanding Expert Problem Solving. *Georgia Institute of Technology, College of Computing Technical Report no. GIT-CC-92/43*. Atlanta, Georgia.

Riesbeck, C. (1993) Replacing CBR: Now What? Invited talk. *AAAI-93 Workshop on Case-Based Reasoning*. Washington, DC, July.

Russel, S., & Norvig, P. (1995). *Artificial intelligence: A modern approach*. Morgan Kaufmann.

Schank, R. & Abelson, R. (1977). *Scripts, plans, goals and understanding*. LEA.

Simina, M. & Kolodner, J. (1995). Opportunistic Reasoning: A Design Perspective. In *Proceedings of the 17th Annual Cognitive Science Conference*. Pittsburg, PA, July 1995.

Smyth, B. & Keane, M. (1995). Remembering to Forget: A Competence-Preserving Deletion Policy for CBR Systems. In *Proceedings of IJCAI-95*.

Sternberg, R. J. (1985). Teaching Critical Thinking: Are We Making Critical Mistakes? *Phi Delta Kappa*, November 1985, p194-198.

Sternberg, R. J. (1986). *Intelligence Applied: Understanding and Increasing Your Intellectual Skills.* Harcourt, Brace, and Jovonovitch.

Turner, R. (1989). A schema-based model of adaptive problem solving. Ph.D. diss., *GIT-ICS-89/42.*, Department of Information and Computer Science, Georgia Tech.

Veloso, M. (1994). *Planning and Learning by Analogical Reasoning.* Springer-Verlag.

Weld, D. (1994). An introduction to least-commitment planning. *AI Magazine, (15) 4,* Winter 1994, pages 27-61. *86/294* Dept. of Electrical Engineering & Computer Science, University of California - Berkeley.

Wilensky, R (1986). Some Problems and Proposals for Knowledge Representation *Technical Report #UCB/CSD 86/294* Dept. of Electrical Engineering & Computer Science, University of California - Berkeley.

Wooldridge, M. & Jennings, N. (1995) Intelligent Agents: Theory and Practice. Submitted to *Knowledge Engineering Review* October 1994, Revised January 1995.